НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)»

Виконав:
студент IV курсу, групи ІО-63
Падучак Дмитро Володимирович _____

Керівник:
асистент кафедри ОТ
Стешин Віктор Васильович _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.
Сімоненко Валерій Павлович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Падучаку Дмитру Володимировичу

1. Тема проєкту «Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)», керівник проєкту Стешин Віктор Васильович, асистент кафедри ОТ, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. технічне завдання

4. Зміст пояснювальної записки дослідження предметної області, огляд існуючих рішень, визначення вимог і завдань для програмного продукту, вибір платформи та технології, обґрунтування оптимальності використання обраних інструментів для розробки, реалізація проєкту.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) схема функціональна, схема принципова, схема структурна

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Відмітки про виконання
1	*Затвердження теми роботи*	01.09.2019	виконано
2	*Вивчення та аналіз завдання*	02.09.2019-02.02.2020	виконано
3	*Розробка архітектури та загальної структури систем*	03.02.2020-03.03.2020	виконано
4	*Розробка структур окремих Підсистем*	04.03.2020-15.03.2020	виконано
5	*Програмна реалізація системи*	16.03.2020-12.04.2020	виконано
6	*Оформлення пояснювальної записки*	13.04.2020-17.05.2020	виконано
7	*Захист програмного продукту*		
8	*Передзахист*	26.05.2020	
9	*Захист*		

Студент Дмитро ПАДУЧАК

Керівник Віктор СТЕШИН

Анотація

У бакалаврській дипломній роботі реалізовано систему автоматичного резервування авіаквитків на базі відкритого API, призначеної для вбудовування її в сайт готелю.

Програмний продукт дозволяє знаходити та резервувати авіаквитки відразу на сайті готелю. Програмний продукт був реалізований на мові JavaScript у візуальному середовищі Visual Studio Code.

Annotation

In this work for a Bachelor's Degree, automatic air ticketing system based on public API is realized designed to embed it in the hotel website.

The software product makes it possible to find and book tickets directly on the hotel's website. The software product was realized in the Javascript language in the Visual Studio Code visual environment.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ.467100.002 ТЗ	Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина). Технічне завдання	4	
3	А4	ІАЛЦ.467100.003 ПЗ	Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина). Пояснювальна записка	66	
4	А4	ІАЛЦ.467100.004 А1	Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина). Схема структурна – структура програми	1	
5	А4	ІАЛЦ.467100.005 А2	Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина). Схема функціональна – схема прецедентів	1	
6	А4	ІАЛЦ.467100.006 А3	Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина). Схема принципова – схема алгоритму пошуку аеропорту	1	

Технічне завдання
до дипломного проєкту

на тему: «Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)»

Київ – 2020

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку клієнтської частини системи автоматичного резервування авіаквитків на базі відкритого API.

Область застосування: альтернатива сучасним системам автоматичного резервування авіаквитків.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання розробки клієнтської частини системи автоматичного резервування авіаквитків на базі відкритого API, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка клієнтської частини системи автоматичного резервування авіаквитків на базі відкритого API.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в періодичних виданнях, публікації в Інтернеті за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Розробка інтерфейсу для пошуку квитка;
- Розробка інтерфейсу для вибору квитка;
- Розробка інтерфейсу для збору даних про пасажира.

5.2. Вимоги до програмного забезпечення

- Операційна система Windows, Linux, macOS
- Остання версія браузерів Chrome, Mozilla

6. ЕТАПИ РОЗРОБКИ

	Дата
Затвердження теми роботи	01.09.2020
Вивчення та аналіз завдання	02.09.2019-02.02.2020
Розробка архітектури та загальної структури систем	03.02.2020-03.03.2020
Розробка структур окремих підсистем	04.03.2020-15.03.2020
Програмна реалізація системи	16.03.2020-12.04.2020
Оформлення пояснювальної записки	13.04.2020-17.05.2020
Захист програмного продукту	
Передзахист	26.05.2020
Захист	

Пояснювальна записка
до дипломного проєкту

на тему: «Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)»

Київ – 2020

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

ОС	Операційна система
Фреймворк	Інфраструктура програмних рішень, що полегшує розробку складних систем.
JS	Мова програмування JavaScript
DOM	Об'єктна модель документа (англ. Document Object Model, DOM) — специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами
HTML	Мова розмітки гіпертекстових документів (англ. HyperText Markup Language) — стандартна мова розмітки веб-сторінок в Інтернеті.
API	Прикладний програмний інтерфейс (англ. Application Programming Interface, API)
JSON	Текстовий формат обміну даними між комп'ютерами, який дозволяє описувати об'єкти та інші структури даних (англ. JavaScript Object Notation)
GDS	(англ. Global Distribution System, GDS) — міжнародна комп'ютерна система бронювання
OTA	(англ. Online Travel Agenc) — Інтернет-туристичне агентство
IATA	(англ. International Air Transport Association) – Міжнародна ассоціація повітряого транспорта
XML	(англ. eXtensible Markup Language) – мова програмування, яка складається з оголошень в вигляді інформації і певних тегів
HTTP	(англ. HyperText Transfer Protocol) – протокол, який дозволяє отримувати різні ресурси, наприклад HTML-документи

URL	(англ. Uniform Resource Locator) – система унікальних адрес електронних ресурсів
БД	база даних
DDL	(англ. Data Definition Language) – одна із конструкцій SQL
DML	(англ. Data Manipulation Language) – одна із конструкцій SQL
DCL	(англ. Data Control Language) – одна із конструкцій SQL

ВСТУП

За останні десятиліття, розвиток систем бронювання онлайн сильно вплинув на туристичну індустрію. Після їх створення, готелі почали включати в свої послуги можливість забронювати не тільки номер, а і екскурсії, транспорт до готелю та різноманітні розваги. Все це можна забронювати на сайті готелю, відразу коли бронюєш номер, цим самим зекономивши час та зусилля на пошук та бронювання всіх послуг окремо.

Зараз готелі в якомусь сенсі навіть конкурують через це, бо люди зазвичай обирають готель де можна більшість послуг забронювати відразу і не витрачати час на їх пошук.

Ця конкуренція підняла нарешті питання про можливість вбудовування системи автоматичного резервування авіаквитків в сайт готелю, щоб люди витрачали на планування поїздок ще менше часу.

Актуальність теми

Виникає потреба у розробці програм, що дозволять витрачати на пошук та резервування різних послуг менше часу, так як в наш час системи онлайн бронювання дуже популярні.

Так як у кожного готелю, в наш час, є сайт, на якому можна забронювати номери онлайн, вбудовування системи автоматичного резервування авіаквитків, допоможе готелю виділитись серед конкурентів.

Мета і задачі дослідження

Метою роботи є розробка клієнтської частини системи автоматичного резервування авіаквитків на базі відкритого API, для вбудовування її в сайт готелю.

Для досягнення поставленої мети були поставлені наступні основні задачі:

- Провести аналіз існуючих систем автоматичного резервування авіаквитків;
- Створити програмну реалізацію клієнтської частини розробленої системи;
- Провести тестування розробленої системи.

Практичне значення

Запропонована клієнтська частина системи автоматичного резервування авіаквитків на базі відкритого API полягає в тому, що її можна із легкістю вбудувати в будь-який сайт готелю.

РОЗДІЛ 1
ОГЛЯД ІДЕЇ СТВОРЕННЯ СИСТЕМИ АВТОМАТИЧНОГО РЕЗЕРВУВАННЯ АВІАКВИТКІВ ВБУДОВАНОЇ В САЙТ ГОТЕЛЮ

1.1. Системи автоматичного резервування авіаквитків

Розвиток та збільшення кількості авіаліній, літаків, а також зростання об’ємів авіаперевезень привели до необхідності створення та використання комп’ютерних систем автоматичного резервування авіаквитків.

Зараз у кожної авіакомпанії є сайт, на якому можна знайти квитки на потрібний рейс і забронювати їх онлайн. На таких сайтах можна обрати дату, час, клас, цінові рамки і безліч інших фільтрів. Але для того щоб знайти сайт авіакомпанії, квитки якої будуть підходити вам найкраще по якимось критеріям, потрібно передивитись безліч сайтів і витратити багато часу та зусиль.

Саме із цієї причини створили сайти, на яких можна знайти та зарезервувати квитки відразу декількох авіакомпаній. Це значно полегшує пошук оптимальних квитків. На таких сайтах зазвичай збирають рейси авіакомпаній які найбільш схожі між собою в цінових рамках та класах перельотів.

Перельоти літаком один із найдорожчих варіантів транспортування. Набагато дешевшими завжди залишаються варіанти пересування різними автобусами, а також поїздами. Але це займає дуже багато часу, тим паче якщо пункт призначення не сусіднє місто і навіть не сусідня країна. Саме тому найкомфортнішим варіантом рахуються авіаперельоти.

Коли сайти, на яких можна знайти та зарезервувати квитки відразу декількох авіакомпаній почали набирати популярність, почали з’являтись лоукост-компанії, які на своїх сайтах пропонують забронювати не просто білети деяких авіаперевізників, а найвигідніші варіанти, різноманітні пропозиції та акції. Саме завдяки таким сервісам, пересування літаком в наш час уже не являється на стільки дорогим варіантом. Зараз полетіти кудись літаком може виявитись навіть дешевше ніж поїхати на автобусі чи

скористуватись іншим видом транспорту. Але якщо спробувати знайти справді найдешевші квитки на визначені дати, то це означає підписатись на нескінченні поштові розсилки, щоденно переглядати різні сервіси де зібрані пропозиції декількох авіакомпаній, встановлювати розширення проти пасток систем бронювання, які відслідковують ваш пошук авіаквитків і з кожним разом піднімають ціну.

До переваг користування сервісами для пошуку та резервування авіаквитків онлайн можна віднести:

1. Миттєвий вибір та подальше резервування авіаквитка на необхідну дату, певний маршрут не покидаючи свого робочого місця або житла. Скрізь, де є доступ до інтернету можна скористатися сервісами для пошуку та резервування авіаквитків.
2. Моментальна оплата за допомогою електронного гаманця або банківської карти.
3. Пасажир дуже швидко отримує інформацію про майбутній переліт, отримавши на вказану при бронюванні електронну адресу маршрутну квитанцію, яку він зможе відразу роздрукувати.
4. Системи пошуку та резервування авіаквитків онлайн працюють щоденно і цілодобово, без вихідних.
5. Виключення можливості втрати квитка, тому що квиток в електронному вигляді.
6. Відсутність касових зборів.
7. Можливість заздалегідь обрати місце в літаку.
8. Доступ до актуальної інформації про перельоти та їх тривалість, дати вильоту та тарифи.
9. Можливість самостійно порівнювати ціни між різними сервісами щоб обрати оптимальний варіант.
10. Придбання квитків відразу на кілька людей без ніяких проблем і потреби особистої присутності.

11. Доступ до інформації про доступні акції, що дозволяють значно зекономити кошти.

Не дивлячись на множину значних переваг користування сервісами для пошуку та резервування авіаквитків онлайн, існують і недоліки, на які теж варто звернути увагу. Серед них:

1. Ймовірність технічного збою в системі резервування авіаквитків, нестабільної роботи програмного забезпечення на стойці реєстрації та інших технічних неполадок.

2. Ймовірність наткнутись на шахраїв.

3. Повернення коштів за невикористаний квиток відбувається не відразу, а часто протягом місяця.

4. При зміні авіакомпанією розкладу, виникає досить багато проблем зі зміною або поверненням квитка самостійно.

5. Багато авіакомпаній взагалі не підтримують можливість резервування квитків онлайн, а це означає, що при користуванні сервісами для пошуку та бронювання авіаквитків в інтернеті, квитки цих авіакомпаній показуватись не будуть і користувач може не знайти оптимального варіанту для себе.

1.1.1 Типи сервісів для пошуку та резервування авіаквитків

Можна виділити основні три типи сервісів для пошуку та резервування авіаквитків, які існують сьогодні:

1. Сайти авіакомпаній

Дозволяють бронювати квитки відразу у авіаперевізників. Перевага такого типу сервісів у тому що при виникненні якихось проблем можна звернутись на пряму до служби підтримки авіакомпанії. Але не дивлячись на те, що це сайт самої авіакомпанії, ціни на квитки в таких сервісах можуть бути навіть дорожчими ніж в агентів.

2. Сайти агентств

На основі укладених договорів з авіакомпаніями, агентства здійснюють діяльність з резервування та продажу авіаквитків. Досить часто можна побачити те, що ціни на квитки на сайтах агентств іноді навіть дешевше ніж на ресурсах самих авіаперевізників. Такі ситуації пояснюються тим, що авіакомпанії, з метою економії та залучення більшого числа потенційних клієнтів, продають квитки агентствам по оптовім тарифам. А вони в свою чергу отримують комісійні, тобто певний відсоток від продажу. Такі сервіси зручні тим, що кожен може побачити всі можливі варіанти перельотів, в тому числі і варіанти перельотів з пересадками, і має можливість самостійно знайти і зарезервувати квитки по оптимальним цінам. Але якщо у вас виникнуть якісь проблеми, вам доведеться мати справу зі службою підтримки не тільки агентства де ви знайшли квиток, але і з самою авіакомпанією.

3. Пошукові агрегатори

Вони не займаються продажом квитків, а здійснюють пошук квитків по міжнародним системам бронювання GDS і просто надають максимально повну інформацію про самі квитки, їх наявність та вартість. Користувачу залишається тільки обрати варіант, який підходить йому найбільше із запропонованих, перейти на сайт авіакомпанії, квиток якої він обрав, і здійснити його бронювання або купівлю.

Один із типів сервісів для пошуку та резервування авіаквитків – метапошукові системи. Вони підключені до Online Travel Agency (OTA) і отримують інформацію про наявність, вартість та всі умови тарифу з цих баз даних. Створений алгоритм формує видачу, відповідно до запиту користувача. Далі є можливість відфільтрувати результати за принципом «найшвидший» або «найдешевший».

Функція метапошуку універсальна – зібрати існуючі пропозиції, показати користувачу та перекинути його на сайт OTA для резервування вибраних квитків.

Ця система збирає дані про рейси прямо від авіакомпаній та інших транспортних перевізників. Формування видачі, яку бачить користувач на

сайті залежить від того як система її отримує. Також можна відображати дані так, як вони представлені у авіакомпаній, а можна придумати більш складну систему, щоб принести користь майбутньому користувачу і цим самим заохотити його на наступний вибір системи резервування авіаквитків.

1.2 Аналіз існуючих систем автоматичного резервування авіаквитків

Аналіз існуючих способів та систем пошуку та резервування авіаквитків дозволив зрозуміти ряд їхніх вагомих недоліків, які занижують ефективність роботи авіакомпаній.

- Дискримінація авіакомпаній власниками систем конкурентів;
- Завищення тарифів при використанні одних глобальних систем;
- Відрахування агентству-посереднику в вигляді відсотків від суми доходів;
- Великі часові розриви між платежем за авіапереліт і появою коштів на рахунку авіаперевізника за надану послугу;
- Існуючі системи не дозволяють авіакомпаніям проводити автоматизований збір статистичних даних про запити авіапасажирів і цим самим формувати керуючі рішення по вдосконаленню процесу авіаперевезень в режимі реального часу.

Відсутність нових ідей в розробці в сфері туризму та авіаперельотів призводить до відставання цієї сфери технологій, а існуючі системи не забезпечують вже необхідною інфраструктурою, заснованою на сучасних телекомунікаційних та комп’ютерних технологіях. В підсумку це приводить до зниження ефективності роботи всього туристичного діла.

1.2.1 Недоліки існуючих систем автоматичного резервування

За останні десятиліття, розвиток систем бронювання онлайн сильно вплинув на туристичну індустрію. Після їх створення, готелі почали включати в свої послуги можливість забронювати не тільки номер, а і екскурсії, транспорт до готелю та різноманітні розваги. Все це можна забронювати на

сайті готелю, відразу коли бронюєш номер, цим самим зекономивши час та зусилля на пошук та бронювання всіх послуг окремо.

Зараз готелі в якомусь сенсі навіть конкурують через це, бо люди зазвичай обирають готель де можна більшість послуг забронювати відразу і не витрачати час на їх пошук. Саме тому кожен власник готелю зараз намагається вбудувати в свій сайт якомога найбільше функцій.

Але не дивлячись на великий обсяг різноманітних послуг які зараз можна зарезервувати на сайті готелю відразу коли бронюєш номер, людям все одно доводиться окремо знаходити та резервувати квитки на транспорт, щоб дістатись до країни та міста призначення зарезервованого готелю. Це виявляється особливо великою проблемою, коли цим транспортом є літак і щоб знайти оптимальний рейс, користувачу доводиться витратити ще досить велику кількість часу та зусиль.

Це підняло питання про можливість вбудовування системи автоматичного резервування авіаквитків в сайт готелю, що ще більше б полегшило онлайн бронювання для людей.

1.3 Створення системи автоматичного резервування авіаквитків вбудованої в сайт готелю.

Ідея вбудовування в сайт готелю системи резервування авіаквитків – це вирішення проблеми витрати зайвого часу та зусиль на пошук та резервування окремо готелю з різноманітними послугами та окремо квитків на літак, які оптимально будуть підходити і по датам, і по місцю розташування аеропортів біля готелю.

Всі люди зазвичай резервують квитки на літак і тільки після цього шукають готель, який максимально буде підходити до заброньованих авіаквитків. Через це користувачі не прив’язуються до конкретного готелю, навіть якщо він їм сподобався і в наслідок – готель втрачає потенційних клієнтів.

Система автоматичного резервування авіаквитків вбудована в сайт готелю, забезпечить готелю зменшення втрати потенційних клієнтів, так як вони будуть бачити на сайті, відповідно до обраних дат, вільні номера готелю та оптимальні рейси які максимально будуть підходити для обраної поїздки.

Це нововведення буде дуже корисним, тому що люди будуть максимально економити свій час на планування різних поїздок. Коли людина шукає квитки на літак, які б підходили до обраних номерів в готелі, їй доводиться шукати рейси на визначені дати, підбирати аеропорт. Система автоматичного резервування авіаквитків вбудована в сайт готелю буде автоматично підбирати рейси, які підходять до обраних дат бронювання номеру та прибувають в аеропорти, що знаходяться неподалік від готелю.

Резервування авіаквитків на сайті готелю призведе до збільшення часу, проведеного користувачами на сайті, а це збільшує вірогідність що користувач може зацікавитись ще якимись додатковими послугами, які надає готель і забронювати ще й їх, а це вже приведе до збільшення прибутку готелю.

Також збільшення часу, проведеного користувачами на сайті готелю призведе до додаткового прибутку через рекламу, яку готель може розміщувати на своєму сайті.

І звичайно ще одна причина, чому вбудовування системи автоматичного резервування авіаквитків в сайт готелю призведе до нового потоку потенційних клієнтів в готель це те, що люди завжди люблять користуватися нововведеннями і не відставати від розвитку нових технологій.

Конкурентоспроможність готелю, в сайт якого буде вбудована система автоматичного резервування авіаквитків відразу зросте, що також є великим плюсом в сфері туристичного бізнесу. Через постійну конкуренцію, найближчим часом всі готелі зацікавляться цією можливістю.

1.4 Аналіз інструментів для створення клієнтської частини системи автоматичного резервування авіаквитків вбудованої в сайт готелю.

Front-end розробка починається з побудови так званого «скелета» сайту. Для того щоб створити цей «скелет» ми скористаємося мовою програмування HTML.

[1]HTML (HyperText Markup Language) – це мова гіпертекстової розмітки, за допомогою якого створюють структуру web-сайту. За допомогою маркерів або як їх ще називають теги (tags), які пишуться в середині ламаних дужок, формують ті елемент, з яких і складається код HTML. Більшість елементів цієї мови мають закриті і відкриті теги. Від правильного (валідного) використання елементів залежить не тільки адекватна структура веб-сторінки, але і правильне відображення її в інтернет-браузері.

Як правило любий HTML код завжди починається з <!DOCTYPE html>. Для браузера наявність цього елемента вказує на те що, йому потрібно відобразити HTML-структуру:

```
<!DOCTYPE html>
<html>
    <body>
        <h1>Тут заголовок</h1>
        <p>Тут абзац</p>
    </body>
</html>
```

Продемонстрований невеличкий варіант простого HTML-кода, який містить в собі декілька елементів, які, в свою чергу, складаються з таких тегів як:

- <body></body> - тег, який відкривається і закривається та вказує на початок і кінець тієї частини сторінки, в якій буде міститься контент сторінки. На українську мову цей тег можна перекласти як «тіло».
- <h1></h1> - вказує на початок і закінчення заголовка. Всього таких тегів може бути 6, і відрізняються вони величиною шрифту – чим вище числовий порядок заголовка, тим менший розмір шрифту. Разом з

тегом , <h1> формує структуру самого контенту: зокрема мітки відображають початок і кінець абзацу тексту.

Таким чином, один за одним формуються різні елементи, які як наслідок будуть відображені на сторінці. Для того щоб указати закриття якогось тегу, в HTML прийнято ставити слеш «/».

Після того як ми написали якусь структуру, вона виглядає так:

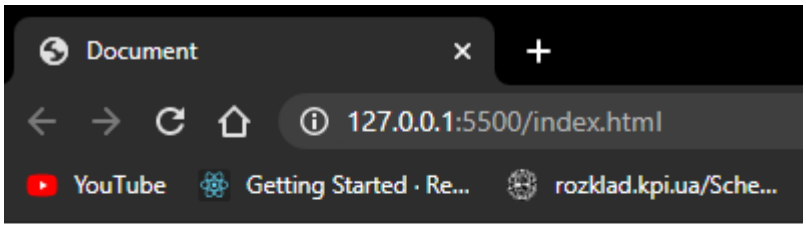


Рис. 1.1 Вигляд структури на HTML

Не дуже гарно. Для того щоб зробити сайт більш естетичним і красивим використовується CSS.

[2] CSS (*Cascading Style Sheets)* – це формальна мова, служить для описання оформлення зовнішнього вигляду документа.

Якщо документ створений з використанням тільки HTML, то в ньому визначається не тільки кожен елемент, але і спосіб його відображення (колір, шрифт і т.д). Якщо підключити до HTML CSS, тоді HTML використовується тільки для визначення порядку об’єктів, а за всі властивості відповідає CSS. В HTML достатньо прописати клас і не перелічувати всі стилі кожен раз.

Така технологія:

- Забезпечує відносно просту і швидку розробку, тому що прописавши стилі один раз, їх можна використовувати багато разів.
- Підвищує гнучкість і зручність редагування – достатньо внести правку в CSS, щоб оформлення змінилось повсюди.

- Робить код більш простим, знижує повторення елементів. Його простіше читати програмістам і легше знаходити неполадки.
- Пришвидшує час завантаження, тому що CSS може кешуватись при першому відкритті, а при наступних зчитуються тільки структура і дані.
- Забезпечує можливість легко застосовувати до одного документу різні стилі.

Тобто всі стилі служать не тільки для відтворення дизайну, але і кардинально змінюють підхід до будівництва сайту, полегшуючи роботу розробника та забезпечуючи гнучкість реалізації. Саме для цього потрібен CSS.

CSS можна охарактеризувати простими словами як набір правил, які описують, як повинен виглядати елемент.

Правило складається з селектора і блока оголошення

Рис. 1.2 Приклад правила

CSS можна зв’язати з HTML декількома способами:

- Добавлення тега <style> з атребутом type="text/css".
- Всередині тега, за допомогою атрибута style. При цьому не потрібно вказувати селектори.
- Підключення зовнішньої таблиці стилів: <link rel="stylesheet" href="шлях до style.css" type="text/css"/>.

Далі після того як в нас є так звана верстка, нам потрібно додати якийсь функціонал. За це відповідає мова JavaScript.

[3] JavaScript – це об’єктно-орієнтована мова програмування, яка містить в собі мову ECMAScript. Часто цю мову програмування використовують для розробки веб-додатків, з метою надання їм цікавості і «живого» ефекту.

Основні функції які виконує JavaScript:

1. Можливість змінювати сторінки в браузері
2. Добавлення або видалення тегів
3. Зміна стилів сторінки
4. Інформація про дії користувача на сторінці
5. Запит на доступ до початкового коду сторінки
6. Внесення змін в цей код

Переваги JavaScript:

- Не один сучасний браузер не обходиться без підтримки JavaScript
- JavaScript – мова, яку з легкістю може вивчити людина, яка ніколи не займалась програмуванням
- Корисні функціональні налаштування
- Постійне вдосконалення та підтримка мови

Після постійних покращень, ця мова JavaScript набула особливого статусу. В сучасному світі Front-end розробка ніяк не може обійтись баз цієї мови.

JavaScript почали настілки часто використовувати, що з’явилися навіть фреймворки.

[4]Фреймворк – це програмний продукт, який полегшує розробку і подальшу підтримку тяжких та великих проектів. Фреймворк, як правило, містить в собі тільки базові програмні модулі, а всі спеціальні компоненти робить сам програміст на основі даної бази. Цим самим досягається не тільки висока швидкість розробки, а й більша продуктивність та надійність.

На даний момент є 3 найбільш популярні Js фреймворки. А саме:

- React
- VueJs

- Angular

[5] React – JavaScript бібліотека з відкритим вихідним кодом, що використовується для розробки інтерфейсу користувача. React розробляється і підтримується Facebook. Яскравим прикладом роботи React є Instagram.

```
<div id="myReactApp"></div>

<script type="text/babel">
  class Greeter extends React.Component {
    render() {
      return <h1>{this.props.greeting}</h1>
    }
  }

  ReactDOM.render(<Greeter greeting="Hello World!" />,
document.getElementById('myReactApp'));
</script>
```

Рис. 1.3 Приклад коду який написаний на React

Особливістю React є односпрямована передача даних. Властивість передачі даних від батьківського компоненту дочірнім. Компонент отримує властивості як множина невідомих значень, тому компонент не може напряму змінювати властивості, але може викликати зміни через callback-функції.

React використовує віртуальний DOM. React створює кеш-структуру в пам’яті, що дозволяє визначити різницю між попереднім і поточним станом інтерфейсу для оптимального обновлення DOM браузера. Таким чином програміст може працювати з сторінкою, рахуючи, що вона оновлюється вся, але бібліотека самостійно вирішує, яким чином компоненти сторінки потрібно оновити.

JavaScript XML – розширений синтаксис JavaScript, який дозволяє використовувати HTML-подібний синтаксис для описання структури інтерфейсу.

Як правило, компоненти написані з використанням JSX, але також є можливість використовувати звичайний JavaScript. JSX нагадує іншу мову, яка була створена в Фейсбук для розширення PHP, XHP.

Методи життєвого циклу дозволяють програмісту запускати код на різних стадіях життєвого циклу компоненту.

Наприклад:

- shouldComponentUpdate – дозволяє зупинити створення компоненту за допомогою повернення false, якщо створення не потрібне.
- componentDidMount – викликається після першого створення компоненту. Часто використовується для отримання даних з віддаленого джерела за допомогою API.
- render – важливий метод життєвого циклу. Кожен компонент повинен мати цей метод. Зазвичай він викликається при зміні даних компоненту для створення даних в інтерфейсі заново.

React Hooks дозволяють використовувати стан і інші можливості React без написання класів.

Постійне використання хуків дозволяє розміщати логіку компонента в функції, яка повторно використовується.

[6] Vue.Js – це JavaScript фреймворк, який вмістив в собі все найкраще що є в React і Angular.

Цей фреймворк був створений однією людиною, тому про подальше вдосконалення цього фреймворку нічого не відомо. Не зважаючи на це, він почав набирати великої популярності.

З React, фреймворк Vue взяв ідею віртуального DOM. З Angular було запозичено two-way data binding.

Рис. 1.4 Приклад коду написаного на Vue

Переваги Vue:

- Бібліотека достатньо проста і функціональна. Для того щоб розібратися в цьому фреймворку, потрібен мінімальний багаж знань.
- Вимоги до стека відсутні, тому Vue.Js можна використовувати в любому проекті.
- Фреймворк займає зовсім мало місця. Це економить час завантаження сторінки.
- Дуже велика швидкість розробки. Завдяки можливості використання будь-яких шаблонів і доступності документації, більшість проблем, які з'являються на етапі розробки, дуже швидко вирішуються.
- Можливість знайти і підключитись до проектів майже будь-якого розробника, хто хоч трішечки знайомий з фронтенд розробкою. Низький поріг входження дозволяє працювати з фреймворком.

[7] Angular – JavaScript-фреймворк з відкритим початковим кодом. Був створений компанією Google. Його ціль – розширення браузерних програм на основі MVC – шаблона, а також полегшення тестування та розробки.

```
<div ng-controller="studentsController as studentsCrtl">
   <div ng-repeat="student in studentsCrtl.students">

       Name: <span class="name">{{student.name}}</span>
       Title: <span class="title">{{student.title}}</span>

   </div>
</div>
```

Рис. 1.5 Приклад коду написаного на Angular

[8] В якості мови шаблонів в Angular використовується HTML. Він розширяється за допомогою директив, які додають в код інформацію про потрібну поведінку. Директиви дозволяють сконцентруватися на розробці логіки і працювати більш продуктивніше. Їх можна використовувати повторно, що також підвищує читабельність коду.

При правильному підході, за допомогою Angular можна швидко розробляти навіть великі програми

В Angular використовується схема MVC, яка розділяє логіку, уявлення і дані програми.

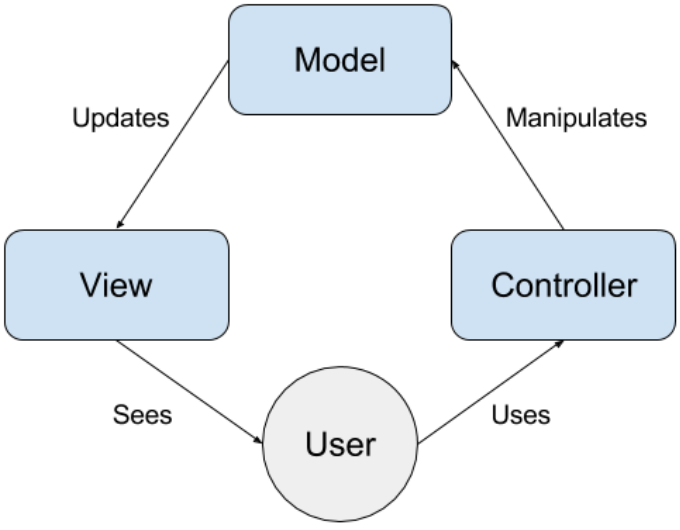


Рис. 1.6 Приклад MVC моделі.

Частини програми знаходяться в середині моделі Angular, ними легко маніпулювати. Таке розбиття на модулі дозволяє завантажувати тільки потрібні служби і ефективно виконувати автоматичне тестування.

ВИСНОВКИ ДО РОЗДІЛУ 1

Розглянувши обсяг доступних для резервування послуг, вбудованих в сайт готелю, можна зробити висновок, що людям все одно доводиться окремо знаходити та резервувати квитки на транспорт, щоб дістатись до країни та міста призначення зарезервованого готелю. Це виявляється особливо великою проблемою, коли цим транспортом є літак. Провівши аналіз сучасних систем автоматичного резервування авіаквитків, можна зробити висновок, що для того, щоб знайти оптимальний рейс, користувачу доводиться витратити ще досить велику кількість часу та зусиль, окремих від пошуку та бронювання готелю. Саме тому була запропонована ідея створення системи автоматичного резервування авіаквитків вбудованої в сайт готелю.

РОЗДІЛ 2
АНАЛІЗ ТЕХНОЛОГІЙ ТА ВИБРАНИХ ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ СИСТЕМИ АВТОМАТИЧНОГО РЕЗЕРВУВАННЯ АВІАКВИТКІВ ВБУДОВАНОЇ В САЙТ ГОТЕЛЮ ТА АЛГОРИТМ ЇЇ РОЗРОБКИ

2.1 Вибрані інструменти

Для вирішення поставленої задачі було вирішено користуватися фреймворком, який тільки починає набирати популярність, – VueJs.

Разом з VueJs, буде використовуватись vue-router, для налаштування переходу між екранами. Також буде використовуватись VueX для передачі даних між клієнтом та сервером, і передачі даних між компонентами.

2.2 Причини вибору VueJs

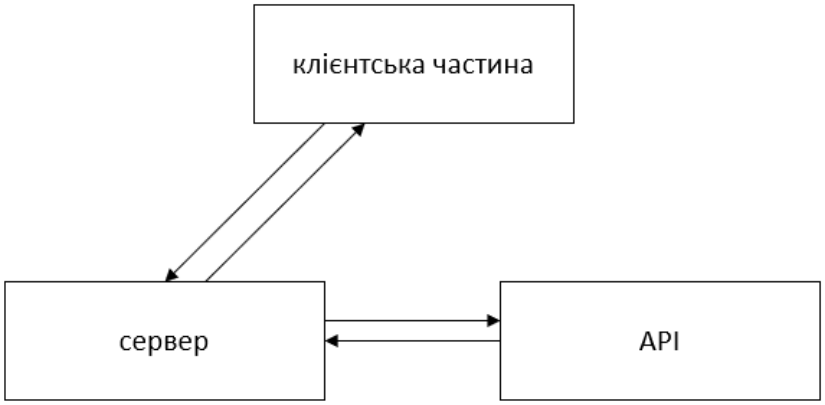
Причини вибору VueJs:

1. Дуже низький поріг входження. Для людини яка трішки розуміється в HTML і CSS, яка може будувати, розробляти макет і задавати просту стилізацію це найкращий вибір. Все що потрібно – це розуміти JavaScript.
2. Так як в VueJs реалізована технологія віртуального DOM, програму можна вставити в будь-яку HTML структуру.
3. Можливість налаштування router.
4. Використання директив, зокрема одна з найкращих директив – v-model, яка реалізує технологію two-way data bindings
5. Компонентний підхід.
6. Дуже легкий функціонал VueX.
7. Займає мало місця.

2.3 Алгоритм розробки програми

2.3.1 Загальний алгоритм розробки програми

Основний алгоритм виглядає так:



На головному екрані користувач вводить дані, після чого вони відправляються на сервер. Після цього сервер обробляє дані і дає відповідь у вигляді масиву квитків.

Після цього користувача переводить на другий екран, де йому показуються всі варіанти квитків. Користувач може детальніше ознайомитись з інформацією про переліт натиснувши кнопку «Details», після чого з’являється модальне вікно, в якому буде вся потрібна інформація. Ще є можливість зарезервувати квиток натиснувши кнопку «BOOK». Також, якщо користувач не натисне кнопку «Details», то він може відразу натиснути кнопку «BOOK» для резервування квитка. Після того, як користувач натиснув кнопку «BOOK» відправляється запит, в тілі якого відправляються всі дані про квиток. А саме «booking_token», після чого отримуємо відповідь про наявність квитка. Якщо все добре, то переходимо на третій екран. Якщо квитка не має в наявності, то повертаємось назад до пошуку квитка.

На третьому екрані користувач повинен заповнити всю інформацію про себе для того, щоб отримати квиток. На цьому екрані є можливість додавання пасажира, якщо натиснути кнопку «ADD passanger». Після того, як користувач натиснув кнопку «ADD passanger» відправляється запит, в якому ми говоримо серверу, що ми хочемо додати користувача, після чого ми отримуємо відповідь, де говориться чи це можливо. Якщо можна – добавляємо, якщо ні – не добавляємо.

Після заповнення всіх даних, користувач натискає кнопку «BOOK» після чого відправляється POST запит з даними.

2.3.2 Вимоги до сторінки пошуку

На сторінці пошуку повинно бути поле для вводу місця, з якого повинен вилітати користувач. Коли користувач вводить хоча б одну букву, відправляється запит на сервер з даними, які вводив користувач. В відповідь отримуємо всі можливі варіанти по заданому запиту. Ці варіанти потрібно розмістити в випадаючому меню, в якому показуються всі можливі варіанти. Ліміт варіантів – 10.

Після того як користувач вибрав потрібний йому варіант, треба заповнити рядок його варіантом. Далі користувач повинен вибрати дату, з якого по яке число програма буде йому шукати всі можливі варіанти.

Є можливість вибору класу перельоту. На даний момент тільки два «Economy» та «Business».Також є можливість вибору типу перельоту чи в одну сторону чи в дві. За замовченням в одну сторону.

Після того, як користувач вибрав переліт в обидві сторони, з’являються ще два поля, в яких він повинен заповнити з якого по яке число програма буде шукати йому варіанти квитка в зворотній бік.

Коли всі данні заповненні, він натискає кнопку «SEARCH NOW». Після чого збираються всі дані і відправляються на сервер.

2.3.3 Вимоги до сторінки квитків

Після того як ми отримали відповідь від сервера – виводимо всі можливі варіанти квитків. В квитку повинні бути:

1. Дата відльоту та прильоту
2. Місце прильоту і відльоту
3. Код місця прильоту і відльоту
4. Час який буде витрачено на переліт
5. Кількість пересадок

6. Клас перельоту
7. Ціна перельоту
8. Кнопка «BOOK»
9. Кнопка «Details» яка викликає модальне вікно

Користувач повинен вибрати квиток, який йому найкраще підходтить. Якщо користувач побачив, що йому вивелись не ті квитки що він хотів, він може повернутися назад до першої сторінки і ввести всі дані знову. Якщо всі дані були вірні і користувач бачить підходящій квиток, але, наприклад, цей переліт з пересадками, він може подивитися весь маршрут перельотів з всіма зупинками натиснувши на кнопку «Details». З'явиться модальне вікно, в якому і буде вся інформація.

Модальне вікно повинне містити:

1. Всі перельоти
2. Місце початкової точки
3. Місця кінцевої точки
4. Загальна ціна
5. Кнопка «BOOK»

Після того як користувач натисне кнопку «BOOK», він переходить на сторінку персональних даних пасажира.

2.3.4 Вимоги до сторінки персональних даних пасажира.

Коли користувач потрапляє на третій екран, він повинен заповнювати свої персональні дані. Користувач може додати ще одного пасажира.

Форма персональних даних пасажира повинна містити:

1. Поле для воду імені
2. Поле для воду прізвища
3. Вибір національності
4. Вибір статі
5. День народження

6. Місяць народження
7. Рік народження

Після того як користувач вів особисті дані, йому потрібно ввести Email і номер мобільного телефону, після чого він натискає кнопку «BOOK».

2.4 Спосіб відправлення даних в VueJs

Основна технологія за допомого якої передають дані називається VueX.

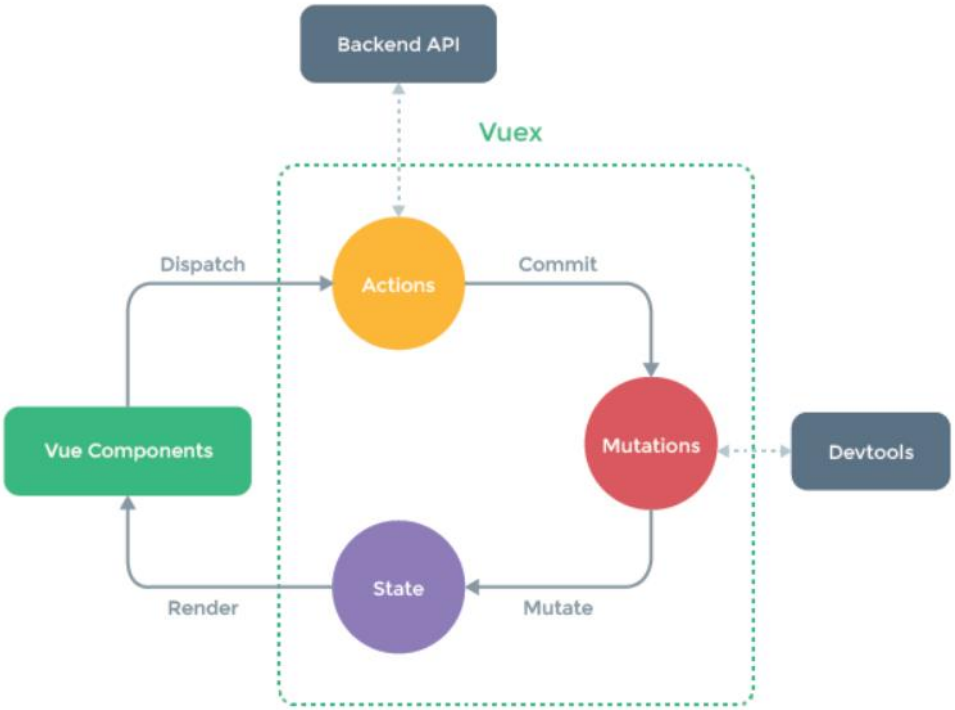


Рис. 2.1 Основна технологія за допомого якої передають дані

Для того що передавати якісь дані нам потрібно створити файл, в який ми будемо передавати всі дані.

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)

export default new Vuex.Store({
  modules: {

  }
})
```

Рис. 2.2 Файл, в який ми будемо передавати всі дані

В файл потрібно імпортувати Vue і також імпортувати Vuex. Потім ми говоримо Vue, що використовуємо Vuex. Після цього нам потрібно створити новий інстанс Vuex, куди в рядок «modules» ми передаємо всі модулі, в яких і буде відбуватися передача даних.

Сам модуль виглядає так:

```
export default {
  actions: {
  },
  mutations: {
  },
  state: {
  },
  getters: {
  },
}
```

Рис. 2.3 Модуль

В рядку actions ми реєструємо функцію, яка і буде передавати дані. Щоб викликати цю функцію потрібно в компоненті імпортувати всі функції які в нас є і використати ту, яка нам потрібна.

```
import {mapGetters, mapActions} from 'vuex'
```

Рис. 2.4 Імпорт функцій з story

Для того, щоб викликати функцію в компоненті нам потрібно записати рядок methods, в середині якого ми імпортуємо, а потім за допомогою this викликаємо функцію:

```
methods: {
  ...mapActions(['test']),
  col(){
    this.test()
  }
},
```

Рис 2.5 Виклик функції з stoty

Після того як ми передали дані в actions, ми передаємо їх mutations.

В mutations ми можемо робити будь-які дії з даними. Після цього, щоб отримати дані ми передаємо їх в state. З state ми вже можемо забирати дані, але краще пропустити їх ще через getters.

2.4.1 Налаштування router

Router – це маршрутизація по програмі. За допомогою router можна переходити між екранами програми. Щоб створити router, потрібно створити окремий файл в якому ми і будемо описувати всі router.

```
1  import Vue from 'vue'  68.1K (gzipped: 24.3K)
2  import Router from 'vue-router'  27.7K (gzipped: 9
3
4  Vue.use(Router)
5
6  export default new Router({
7    mode: 'history',
8    routes: [
9      {
10       path: '/',
11       component: Test
12     }
13   ]
14 })
15
```

Рис. 2.6 Приклад створення router

Потрібно імпортувати Vue і Router. Потім говоримо Vue використовувати Router. Потім створюємо новий істанс Router, в якому і будемо описувати всі router.

В списку routes, описуємо всі шляхи. В path говоримо при якому path буде підгружатись компонент який ми пишемо в component.

2.5 Коротке технічне завдання по створенню клієнтської частини системи автоматичного резервування авіаквитків вбудованої в сайт готелю

Дану систему автоматичного резервування авіаквитків вбудованої в сайт готелю, потрібно розробляти відповідно наступному технічному завданню:

1. Структура проекту
    a. Перший екран, на якому користувач вводить всі дані для пошуку
    b. Квиток
    c. Модальне вікно з інформацією про рейс
    d. Другий екран, в якому користувач вводить всі свої дані для бронювання авіаквитка
    e. Форма особистих даних
2. Перший екран, склад екрана
    a. Поле для вводу місця відправлення
    b. Поле для вводу дати вильоту
    c. Меню для вибору типу перельоту (в одну або в обидві сторони)
    d. Якщо користувач вибрав тип в обидві сторони, то повинні з’являтись поля для вводу дати повернення
    e. Меню для вибору валюти
    f. Меню для вибору класу перельоту
    g. Кнопка пошуку
3. Що потрібен містити квиток
    a. Дата відправлення
    b. Час відправлення
    c. Місце відправлення
    d. Код місця відправлення
    e. Час польоту
    f. Кількість пересадок
    g. Час прильоту
    h. Місце прильоту
    i. Дата прильоту

j. Клас перельоту
k. Вартість квитка
l. Кнопка замовлення
m. Кнопка, яка відкриває модальне вікно з інформацією про переліт

4. Модальне вікно
    a. Повинна бути інформація про кожен переліт
    b. Для кожного перельоту своя карточка
    c. Карточка повинна мати всю інформацію квитка
    d. Кнопка, яка закриває модальне вікно
    e. Ціна перельоту
    f. Кнопка замовлення квитка
5. Другий екран
    a. Форма особистих даних
    b. Поле вводу Email
    c. Поле вводу контактного телефону
    d. Кнопка для того щоб додати ще одного пасажира
    e. Кнопка продовжити
    f. Чекбокс для підтвердження використання особистих даних
6. Форма особистих даних

Якщо користувач не заповнив якісь дані і натиснув кнопку продовжити, потрібно підсвітити ті поля, які не заповненні та вказати що вони обов’язкові для заповнення.

Табл. 2.4 Заповнення форми особистих даних

Елемент	**Обов’язкова для заповнення**	**Коментар**
Поле «Ім’я»	Так	після вводу повинна проводитись валідація для перевірення чи в полях введені тільки букви

Елемент	Обов’язкова для заповнення	Коментар
Поле «Прізвище»	Так	після вводу повинна проводитись валідація для перевірення чи в полях введені тільки букви
Випадаюче меню «Стать»	Так	
Випадаюче меню «місяць народження»	Так	після вводу повинна проводитись валідація для перевірення чи в полях введені тільки числа
Поле «день народження»	Так	після вводу повинна проводитись валідація для перевірення чи в полях введені тільки числа
Поле «рік народження»	Так	після вводу повинна проводитись валідація для перевірення чи в полях введені тільки числа
Випадаюче меню «Вікова група»	Так	

ВИСНОВОК ДО РОЗДІЛУ 2

У цьому розділі були описані вибрані інструменти для розробки клієнтської частини системи автоматичного резервування авіаквитків. Був наведений ряд переваг фреймворку VueJs і причини його вибору для створення даної системи.

Було складено всі вимоги до функціоналу і наведено алгоритм розробки програми, вимоги до сторінки пошуку, сторінки квитків та сторінки персональних даних пасажира.

РОЗДІЛ 3
РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ СИСТЕМИ АВТОМАТИЧНОГО РЕЗЕРВУВАННЯ АВІАКВИТКІВ ВБУДОВАНОЇ В САЙТ ГОТЕЛЮ

3.1 Підготовка середовища до роботи

Для початку, почнемо з того, що встановимо VueJs. Скористаємось заздалегідь встановленим пакетним менеджером npm.

Встановлюємо Vue CLI – стартовий шаблон. Для цього потрібно глобально його встановити на комп'ютер.

```
PS D:\test> npm install -g @vue/cli
```

Рис. 3.1 Команда встановлення Vue CLI

Після того, як встановили шаблон, перевіряємо чи він дійсно встановився.

```
PS D:\test> vue --version
@vue/cli 4.3.1
```

Рис. 3.2 Команда для перевірки наявності Vue

Після введення команди, нам повинно показати ту версію Vue, яку використовує комп'ютер.

Після встановлення всіх потрібних пакетів, проводимо ініціалізацію проекту.

```
PS D:\test> vue create test
```

Рис. 3.3 Команда ініціалізації проекту

Замість слова test ми вводимо назву проекту. Після цього треба відповісти на декілька питань, щоб створити такий проект, який нам потрібно.

В першому питанні треба вибрати чи потрібно додатково встановлювати стандартний набір модулів чи вибрати модулі самому. Ми вибираємо більш розширине встановлення.

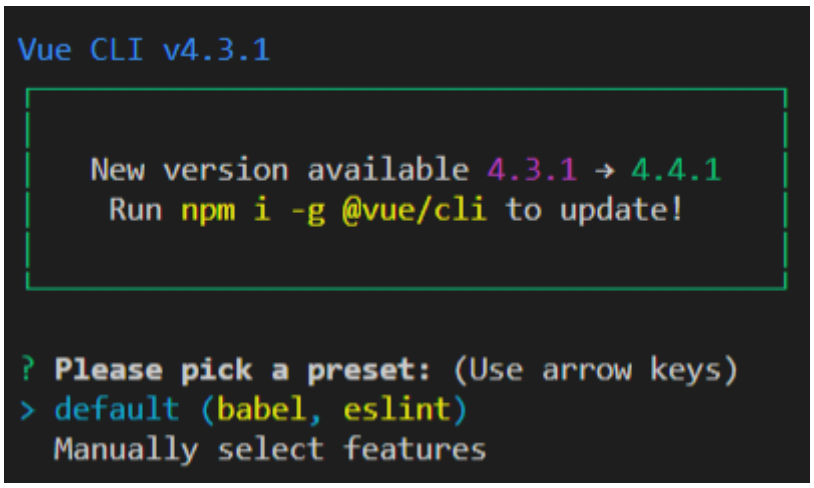


Рис. 3.4 Перше питання Vue CLI

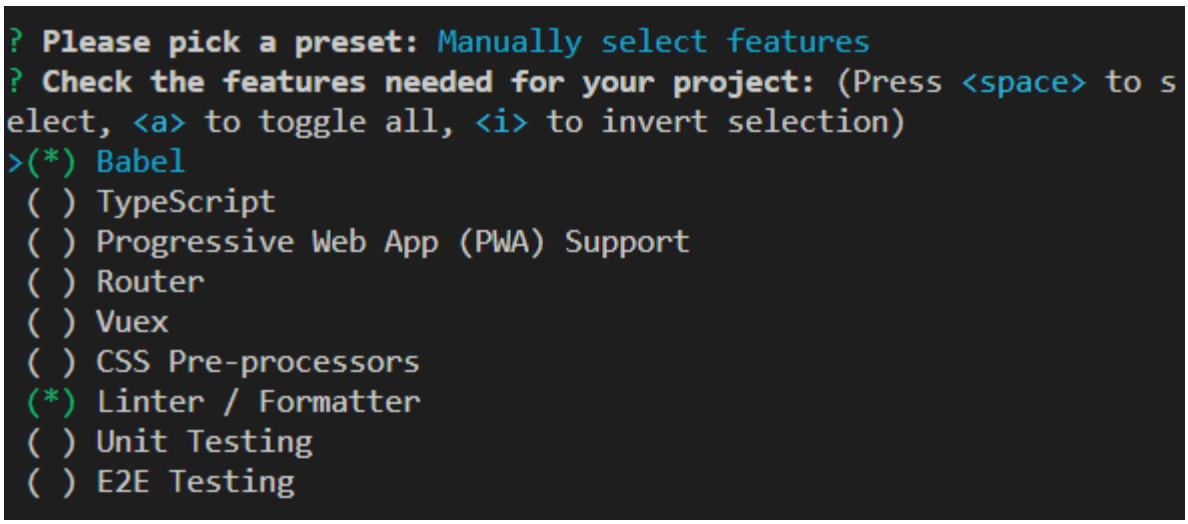


Рис. 3.5 Друге питання Vue CLI

Далі ми вибираємо які модулі ми встановити (Рис. 3.5)

1. [9] Babel – це безкоштовний JavaScript транскомпілятор з відкритим кодом, який в основному використовується для перетворення кода ECMAScript 2015+ (ES6+) в сучасну версію JavaScript, яка може бути запущена старими механізмами JavaScript.
2. [10] Progressive Web App (PWA) Support – технологія в веб розробці, яка візуально і функціонально трансформує сайт в додаток (мобільний додаток в браузері).
3. Router – бібліотека Vue, яка відповідає за маршрутизацію сайту.
4. [11] Vuex – бібліотека Vue, яка служить як центральне сховище даних для всіх компонентів програми з правилами.

5. CSS Pre-processors – препроцесори SASS(SCSS), які полегшують написання CSS.
6. [12] Linter / Formatter – це інструмент аналізу статичного коду для визначення проблемних зразків, знайдених в коді JavaScript.
7. Unit Testing, E2E Testing – це модулі для тестування, в даній роботі вони не потрібні.

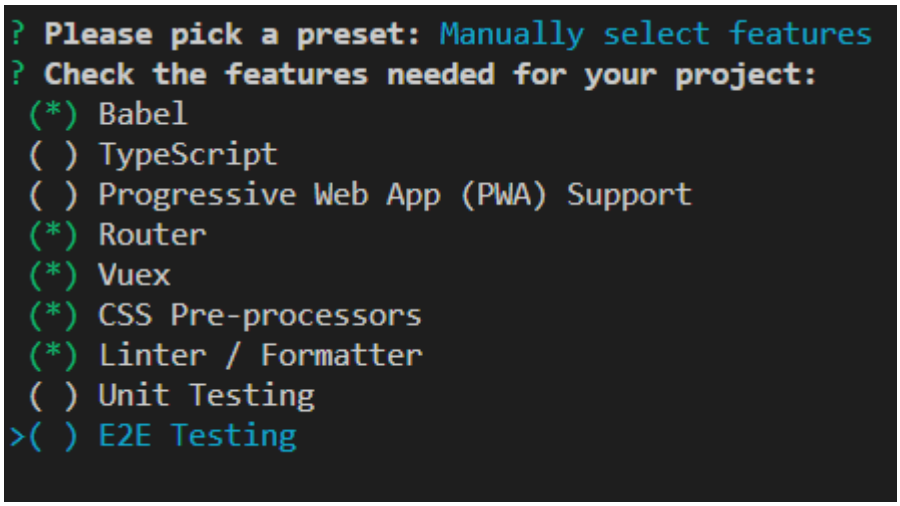


Рис. 3.6 Вибранні модулі

Після того як ми вибрали потрібні модулі починається встановлення шаблону.

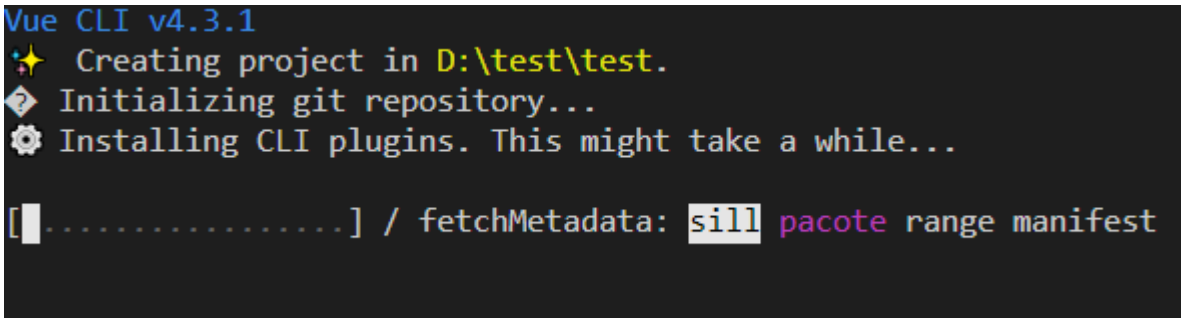


Рис. 3.7 Встановлення шаблону

Після того як все встановиться, потрібно перейти в директорію нашого проекту.

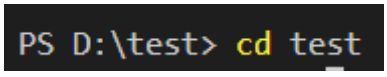


Рис. 3.8 Команда для переходу в директорію

Щоб запустити наш проект потрібно вести:

```
PS D:\test\test> npm run serve
```

Рис. 3.9 Команда для запуску проекту

```
 INFO  Starting development server...
98% after emitting CopyPlugin

 DONE  Compiled successfully in 2551ms


  App running at:
  - Local:   http://localhost:8081/
  - Network: http://192.168.0.112:8081/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

Рис. 3.10 Результат запуску

Після запуску, запускається локальний сервер, в якому і буде відображатися проект.

3.2 Створення компонента

Все в VueJs будується за допомогою компонентів.

Компонент – це частина JavaScript і HTML, з якоюсь логікою, яку можна повторно використовувати.

Компонент в VueJs виглядає так:

```
<template></template>
<script></script>
<style lang="scss"></style>
```

Рис. 3.11 Вигляд компоненту в VueJs

В середині тегу template пишемо основний HTML код.

В середині тегу script пишемо основний функціонал, який буде в цьому компоненті.

В середині тегу style пишемо всі стилі які відносяться до цього компоненту.

Щоб створити компонент потрібно створити файл, який зазвичай називають з великої букви і назва файлу це назва компоненту.

3.3 Створення сторінки пошуку

Для початку нам потрібно створити всі необхідні компоненти:

1. Header
2. Footer
3. Search

Після створення всіх необхідних компонентів, ми імпортуємо всі компоненти в головний компонент. Після всіх імпортів верстка компонента виглядає так:

```
<template>
  <div>
    <Header />
    <Serch />
    <Footer />
  </div>
</template>
```

Рис. 3.12 Вигляд компонента сторінки пошуку

3.3.1 Створення Header

Почати потрібно з того, що треба написати HTML код нашого компонента (див. Рис. 3.13).

Особливість цього компонента в тому що він імпортує в собі ще один компонент Select, який відповідає за випадаюче меню. Після того як написали всю верстку, ми робимо стилізацію.

```vue
<template>
  <header class="header">
    <div class="container">
      <div class="header_body">
        <div class="header__logo">
          AIRWAVE
        </div>
        <div class="header__currency">
          <p>Currency :</p>
          <Select />
        </div>
        <div class="header__bag">
          <img src="../assets/img/header__bag.svg" alt="">
          <p>Cart</p>
          <span class="header__bag-order">1</span>
        </div>
      </div>
    </div>
  </header>
</template>
```

Рис. 3.13 Верстка Header

Після того як, ми зробили всі необхідні дії, наш компонент має такий вигляд:

Рис. 3.14 Вигляд Header

3.3.2 Створення Footer

Цей компонент використовується на всіх сторінках, тому ніяких особливостей в ньому не має.

```vue
<template>
  <footer class="footer">
    <div class="container">
      <div class="footer_body">
        <div class="footer__compani">
          <img src="../assets/img/footer_compani.svg" alt="">
          2011-2020. All rights reserved
        </div>
        <div class="footer__logo">
          AIRWAVE
        </div>
        <div class="footer__phone">
          <img src="../assets/img/footer__phone.svg" alt="">
          + 38 (097) 26-63-235
        </div>
      </div>
    </div>
  </footer>
</template>
```

Рис. 3.15 Верстка Footer

Рис. 3.16 Вигляд Footer

Після створення всіх потрібних компонентів наша сторінка виглядає так:

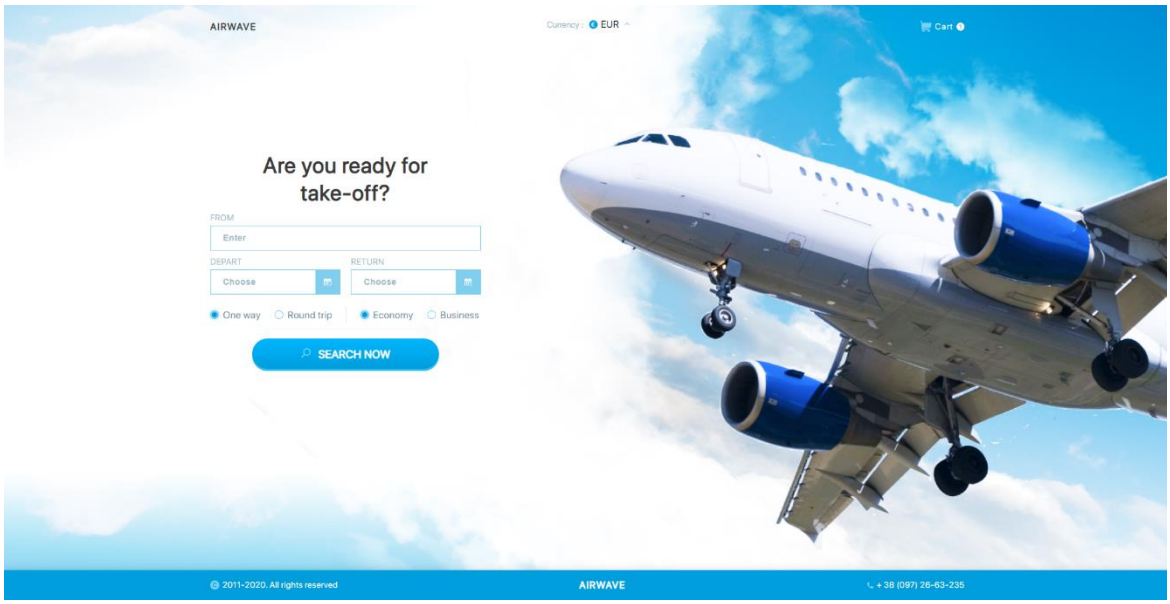


Рис. 3.17 Вигляд сторінки пошуку

3.4 Реалізація пошуку квитка

Реалізація пошуку починається з того, що нам потрібно створити компонент, в який будемо дозавантажувати всі інші компоненти.

```
<template>
  <div class="container-fluid">
    <form class="serch">
      <div class="container">
        <div class="serch__body">
          <h1 class="serch__title">
            Are you ready for take-off?
          </h1>
          <div class="serch-container">
            <ImputBlockCityFrom
            :inputBlockData="inputData_inputBlock_From"/>
          </div>
          <div class="data-block">
            <InputData
              v-for="inputBlockData of inputData_inputDataBlock"
              :inputBlockData="inputBlockData"
              :key="inputBlockData.id"
            />
          </div>
          <div class="checkbox-block">
            <CheckBox
              v-for="checkboxData of checkbox_data"
              :checkboxData="checkboxData"
              :key="checkboxData.id"
            />
          </div>
          <div class="serch__btn">
            <router-link to="/tickets" >
              <img src="../assets/img/serch.svg" alt="">
              SEARCH NOW
            </router-link>
          </div>
        </div>
      </div>
```

Рис. 3.18 Верстка компонента Search

Як видно, ми підключаємо дуже багато інших компонентів і ці компоненти повторюються. І так ми бачимо що за допомогою директиви Vue ми можемо створювати унікальні компоненти.

Щоб передати якісь дані в компонент, скористалися директивою :v-bing, яку ми також можемо спостерігати в скороченому вигляді як просто : .

```
<template>
  <div class="input-block" :id="inputBlockData.idInput">
    <span>{{inputBlockData.lable}}</span>
    <input type="text"
      :placeholder="inputBlockData.placeholder"
      @input="search()"
      data-value
    >
    <div class="serch-result active" v-if="all_airport__from.length">
      <SerchResult
        v-for="data of all_airport__from"
        :serch_result__data="data"
        :key="data.id"
        v-on:sitiCod="log"
      />
    </div>
  </div>
</template>
```

Рис. 3.19 Приклад компонента в який передають дані

За допомогою {{data}} ми говоримо vue які саме дані, а найголовніше де їх потрібно розмістити.

3.4.1 Створення декількох компонентів за допомогою директиви v-for

Важливо те, що ми використовуємо директиви Vue.js для того щоб створити унікальний компонент. За допомогою директиви v-for ми говоримо Vue, що нам потрібно запустити цикл, який пройдеться по масиву і на кожен елемент цього масиву створить свій власний компонент зі своїми даними.

```
checkbox_data: [
  {text: 'One way', checked: 'true'},
  {text: 'Round trip', checked: 'false'},
  {text: 'Economy', checked: 'true'},
  {text: 'Business', checked: 'false'},
]
```

Рис. 3.20 Приклад масиву для створення компонентів

Коли ми запускаємо цикл на першій ітерації, він проходиться по першому рядку і передає дані компоненту. Далі компонент приймає ці дані і розміщає на задані нами раніше місця.

```
<div class="checkbox">
  <label class="checkbox__lable">
    <input type="checkbox" class="checkbox__input" v-if="checkboxData.checked == 'false'">
    <input type="checkbox" class="checkbox__input" checked v-else >
    <div class="checkbox__fake"></div>
    <div class="checkbox__text">
      {{checkboxData.text}}
    </div>
  </label>
</div>
```

Рис. 3.21 Приклад розміщення даних

Після того як відпрацював цикл, можна побачити, що ми створили 4 компонента не просто копіюючи їх і передаючи кожному компоненту свої дані, а написавши одну директиву яка зробить це все за нас.

Рис. 3.22 Приклад відпрацювання циклу

3.4.2 Реалізація пошуку міста

Як видно на Рис. 3.19, ми використовуємо ще одну директиву Vue. @(скорочення v-on) – ця директива, в стандартному JavaScript, виступає як функція on(), яка добавляє обробку подій на якийсь елемент, тобто тіло функції спрацьовує при якійсь події.

В даному випадку, ця функція спрацьовує коли ми щось вводимо в рядок. В свою чергу, ця функція викликає ще одну функцію, яка відсилає за допомогою VueX запит на сервер.

```
...mapActions(['serch_arport__from']),
search(){
  this.serch_arport__from(this.$el.querySelector('input').value)
},
```

Рис. 3.23 Функція, яка відправляє дані в store

3.4.3 Обробка даних в store

Після того, як функція відправила дані в store нам потрібно їх обробити.

```
export default {
  actions: {
    async serch_arport__from(ctx,siti){
      const res = await fetch(`http://localhost:8080/${siti}`)
      const result_airport = await res.json()
      ctx.commit('update_airport', result_airport.locations)
    }
  },
  mutations: {
    update_airport(state, result_airport) {
      state.result_airport = result_airport
    }
  },
  state: {
    result_airport: []
  },
  getters: {
    all_airport__from(state){
      return state.result_airport
    },
  }
}
```

Рис. 3.24 Функціонал store для пошуку міста

Ми приймаємо дані, які були передані функцією, і за допомогою функції fetch посилаємо асинхронний запит на сервер. Після чого ми обробляємо відповідь.

Спочатку ми передаємо відповідь в метод mutations. Для цих даних нам не потрібна проміжна обробка, тому передаємо їх в state. Після чого ми можемо їх забирати, але краще передати їх в getters. В getters ми можемо ці дані якось змінювати, доповнювати, якщо це звичайно потрібно, тому краще їх забирати з getters.

3.4.4 Відображення відповіді сервера

Після того, як ми отримали відповідь і, по потребі, її обробили, нам потрібно її забрати з store і розмістити на сайті.

```
...mapGetters(['all_airport__from'])
```

Рис. 3.25 Функція для отримання даних з store

```
<SerchResult
  v-for="data of all_airport__from"
  :serch_result__data="data"
  :key="data.id"
  v-on:sitiCod="log"
/>
```

Рис. 3.26 Відображення даних

Для відображення, використовуємо вже відому нам директиву v-for.

3.4.5 Ралізація select компонента

Для того щоб користувач міг вибрати потрібну йому валюту, нам потрібно зробити випадаюче меню, в якому він зможе вибрати одну із можливих.

```
<template>
  <div class="select">
    <div class="select__title">{{selectBlockData.title}}</div>
    <div class="select__container" v-on:click = "clickSelect">
      <div class="select__body">
        <div class="select__value placeholder">
          <span>{{selectBlockData.placeholder}}</span>
          <i class="fas fa-angle-up"></i>
        </div>
        <SelectOptions
          v-for="options of selectBlockData.options"
          :options="options"
          :key="options.id"
        />
        <input type="text" class="select__input">
      </div>
    </div>
  </div>
</template>
```

Рис. 3.27 Верстка select компонента

В пасивному стані select виглядає так:

Рис. 3.28 Пасивний стан select

Для того щоб задати select активний стан, ми скористаємося вже знайомою директивою v-on, яка при кліку запускає функцію clickSelert.

```
clickSelect(e) {
    this.$el.classList.toggle('active')
    this.$el.addEventListener('mouseleave', () => this.$el.classList.remove('active'))
    this.$el.querySelectorAll('.select__options').forEach(options => {
        if(e.target == options){
            this.$el.querySelector('.select__value').classList.remove('placeholder')
            this.$el.querySelector('.select__value span').innerHTML = `${options.innerHTML}`
            this.$el.querySelector('.select__input').value = options.dataset.value
            this.$el.classList.remove('active')
        }
    })
}
```

Рис. 3.29 Вигляд функції clickSelert

Ця функція спочатку добавляє клас «active» нашому select. Після чого select переходить в активний стан.

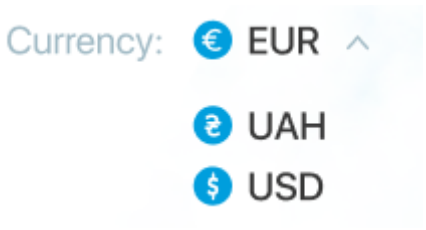


Рис. 3.30 Активний стан select

Потім ми обробляємо подію, коли курсор виходить за рамки нашого select, тоді ми переводимо його в пасивний стан.

Після цього, ми обробляємо подію кліку на дочірній елемент. При кліку на дочірній елемент, ми міняємо значення select на дочірній.

3.4.6 Збір даних і відправлення запиту на пошук квитка

Після того, як користувач вибрав потрібний пункт відправлення, вибрав клас перельоту, тип перельоту і валюти, він хоче побачити всі варіанти перельотів. Для цього потрібно зібрати і відправити всі дані на сервер. Для того, щоб зібрати дані, ми поміщаємо їх всі в один масив, а допоможе в цьому

VueX. Для цього потрібно створити в store нову функцію, яка буде збирати дані.

```
export default {
  actions: {
    async add__all_data(ctx,data){
      ctx.commit('update_data_sarch', data)
    }
  },
  mutations: {
    update_data_sarch(state, data) {
      state.data_sarch.push(data)
    }
  },
  state: {
    data_sarch: []
  },
  getters: {
    all_data_sarch(state){
      return state.data_sarch
    },
  }
}
```

Рис. 3.31 Функція для збору даних

Вона приймає в себе вхідні дані і добавляє їх в масив. А за допомогою функції all_data_search ми зможемо отримати ці дані. Після того, як користувач натискає кнопку «SEARCH NOW» ми запускаємо функцію, отримуємо дані і відправляємо їх на сервер.

```
export default {
  actions: {
    async serch_tickets(ctx,data_tickets){
      const res = await fetch(`http://localhost:8080/tikets/${data_tickets}`)
      const result_tickets = await res.json()
      ctx.commit('update_serch_tickets', result_tickets)
    }
  },
  mutations: {
    update_serch_tickets(state, result_tickets) {
      state.result_tickets = result_tickets
    }
  },
  state: {
    result_tickets: []
  },
  getters: {
    all_result_tickets(state){
      return state.result_tickets
    },
  }
}
```

Рис. 3.32 Функція, яка відправляє запит для пошуку квитка

Після того, як ми отримали відповідь, ми можемо перейти на іншу сторінку.

3.4.7 Перехід на сторінку квитка

Для того щоб перейти на наступну сторінку, ми скористаємося допоміжною бібліотекою Vue-router.

Щоб перейти на наступну сторінку нам потрібно вставити спеціальний компонент.

```
<router-link to="/tickets" >
  <img src="../assets/img/serch.svg" alt="">
  SEARCH NOW
</router-link>
```

Рис. 3.33 Компонент для переходу на іншу сторінку

Спеціальний атрибут to, вказує на яку сторінку нам потрібно перейти.

В головному файлі vue-router ми вказуємо при якому URL який компонент ми підгружаємо.

```
export default new Router({
  mode: 'history',
  routes: [
    {
      path: '/',
      component: Program
    },
    {
      path: '/tickets',
      component: () => import('./views/TicketsBody.vue')
    },
    {
      path: '/passanger',
      component: () => import('./views/PassangerScreen.vue')
    }
  ]
})
```

Рис. 3.34 Файл для вказання router

3.5 Реалізація сторінки квитка

Починаємо реалізацію з того, що робимо головний компонент.

```
<template>
  <div>
    <HeaderBack
      :routData="routData" />
    <div class="ticket-screen">
      <div class="ticket-modal">
        <TiketsModal
        :modal_info_data="modal_info"
        />
        <div class="ticket-modal__bg"></div>
      </div>
      <div class="ticket-screen-title">…
      </div>
      <div class="tickets-container">
        <Ticket
          v-for="tikets of tikets_data"
          :tikes_info="tikets"
          :key="tikets"
        />
      </div>
    </div>
    <Footer />
  </div>

</template>
```

Рис. 3.35 Головний компонент сторінки квитка

В цей компонент ми підгружамо компонент модального вікна і компонент квитка. Ми передаємо дані про квиток в компонент.

3.5.1 Реалізація квитка

Ми запускаємо цикл і він створює всі варіанти квитків. Верстка квитка виглядає так:

```
<template>
  <div class="ticket">
    <div class="ticket__body">
      <div class="ticket__informations_column">
        <div class="ticket__informations">…
      </div>
      <div class="ticket__date">…
      </div>
    </div>
      </div>
      <div class="ticket__buttom">
        <div class="ticket__cost">{{tikets.cost}}</div>
        <div class="ticket__book"

        >BOOK</div>
      </div>
    </div>
    <div class="ticket__more"
      @click="modal_data"
    >…
    </div>
  </div>
</template>
```

Рис. 3.36 Верстка квитка

Важливо, що коли ми натискаємо на клас ticket__more викликається функція modal_data.

```
    modal_data: function(){
        this.$emit('add_class', this.modal_info)
    }
}
```

Рис. 3.37 Функція modal_data

Ця функція передає дані в батьківський елемент і викликає подію, яка відкриває модальне вікно.

3.5.2 Реалізація модального вікна «Більше інформації»

Верстка модального вікна виглядає так:

```
<template>
  <div class="tikets-modal">
    <div class="tikets-modal__title">iTINERARY DETAILS</div>
    <div class="tikets-modal__close"></div>
    <div class="tikets-modal-information">
      <div class="tikets-modal-information__title">…
      </div>
      <div class="tikets-modal-information__container-inf">
        <TiketsModalInformation
          v-for="modal_information of tikets_modal_infirmation"
          :modal_information="modal_information"
          :key="modal_information"
        />
      </div>
    </div>
    <div class="tikets-modal__footer">…
    </div>
  </div>
</template>
```

Рис. 3.38 Верстка модального вікна

В цьому компоненті ми відображаємо всю інформацію про перельоти. Для того щоб відобразити інформацію ми використаємо директиву v-for .

3.5.3 Компонент для відображення інформації в модальному вікні

Після того, як компонент отримує дані, ми їх відображаємо. Скористаємося конструкцією {{ }}.

```
<template>
  <div class="tikets-modal-information__body">
    <div class="tikets-modal-information__date">
      <img src="../../assets/img/ticket/calendar.svg" alt="">
      <span>{{modal_information.deta_siti_from}}</span>
    </div>
    <div class="tikets-modal-information__container">…
    </div>
    <div class="tikets-modal-information__date">
      <img src="../../assets/img/ticket/calendar.svg" alt="">
      <span>{{modal_information.deta_siti_to}}</span>
    </div>
  </div>
</template>
```

Рис. 3.39 Відображення інформації в модальному вікні

3.5.4 Header для повернення до пошуку

Якщо користувач вирішить шукати якийсь інший квиток, йому потрібно буде лише в header написнути кнопку «Back to search», і його поверне на сторінку пошуку. Це реалізовано за допомогою royret.

```
<template>
  <header class="header headerBack">
    <div class="container">
      <div class="header_body">
        <div class="header__back">
          <router-link :to="routData.rout">
            <img src="../assets/img/arrow__gray.svg" alt="">
            <span>BACK TO {{routData.text}}</span>
          </router-link>
        </div>
        <div class="header__logo">
          AIRWAVE
        </div>
        <div class="header__bag">
          <img src="../assets/img/header__bag.svg" alt="">
          <p>Cart</p>
          <span class="header__bag-order">1</span>
        </div>
      </div>
    </div>
  </header>
</template>
```

Рис. 3.40 Header для повернення на попередню сторінку

3.5.5 Перехід на сторінку пасажира

Після того як користувач вибрав підходящий квиток, він натискає кнопку «BOOK», після чого запускається функція яка передає bookig_tooken в запит.

```
export default {
  actions: {
    async ticket_availability(ctx,bookig_tooken,passnger){
      const res = await fetch(`http://localhost:8080/tikets/book/${bookig_tooken}/${passnger}`)
      const result_ticket_availability = await res.json()
      ctx.commit('update_ticket_availability', result_ticket_availability)
    }
  },
  mutations: {
    update_ticket_availability(state, result_ticket_availability) {
      state.result_ticket_availability = result_ticket_availability
    }
  },
  state: {
    result_ticket_availability: []
  },
  getters: {
    all_result_tickets(state){
      return state.result_ticket_availability
    },
  }
}
```

Рис. 3.41 Запит на перевірку наявності квитка

Цей запит потрібен для того, щоб перевірити наявність квитка. В запит ми ще передаємо кількість пасажирів, на перший запит ми передаємо 1 пасажира.

Після того як приходить відповідь, ми запускаємо функцію, яка на протязі 30 хвилин з інтервалом в 15 секунд посилає запити на перевірку наявності квитка.

```
...mapActions(['ticket_availability']),
search(){
  let ticket_availability_interval = setInterval(() => {
    this.ticket_availability(this.bookig_tooken, this.passanger) }, 15000)
  setTimeout(() => { clearInterval(ticket_availability_interval); }, 1800000);
},
```

Рис. 3.42 Функція для запиту кожних 15 хвилин

3.6 Реалізація сторінки пасажира

Для початку створюємо головний компонент.

```vue
<template>
  <div>
    <HeaderBack :routData="routData" />
    <div class="passanger-container">
      <div class="passanger-container__body">
        <Passanger
        v-for="pasetger of pasetger_data"
        :pasetger_data='pasetger'
        :key="pasetger"
        />
      </div>
      <div class="passanger-container__details">...
      </div>
      <div class="passanger-container__btn">
        <div class="passanger-container__add-passanger"
          @click="add_passanger"
        >
          + ADD passanger
        </div>
        <div class="passanger-container__book">
          BOOK
          <img src="../assets/img/arrow__white.svg" alt="">
        </div>
      </div>
    </div>
    <Footer />
  </div>
```

Рис. 3.43 Верстка головного компонента

Важливо, що в нас також є можливість повернутися назад.

В даному випадку v-for проходить одну ітерацію, але в подальшому, якщо змінити список по якому проходить цикл, він виведе стільки компонентів, скільки буде в списку. Це нам знадобиться коли ми захочемо додати пасажира.

3.6.1 Реалізація форми персональних даних

Для початку давайте подивимя на верстку

```
<template>
  <div class="passanger">
    <div class="passanger__title">
      <img src="../../assets/img/passanger.svg" alt="">
      <span>Primary passanger</span>
    </div>
    <div class="passanger__informations">
      <img src="../../assets/img/info.svg" alt="">
      <span>Use all given names and surnames exactly as they appear in your
    </div>
    <div class="passanger__FIO">…
    </div>
    <div class="passanger__row">…
    </div>
  </div>
</template>
```

Рис. 3.44 Верстка форми персональних даних

Це звичайний компонент, який ми будемо використовувати в подальшому.

3.6.2 Реалізація додавання ще одного пасажира

Після того, як ми натиснемо на кнопку «додати пасажира», повинна запуститися функція add_passanger. Ця функція робить наступне, міняє кількість пасажира на 2 і перестворює компонент Passanger, перед тим зберігши всі данні ,які були заповненні на той випадок, якщо було щось введено. І тоді за допомогою директиви v-for ми додаємо новий компонент.

ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі було проведено підготовку середовища до роботи а також була розроблена клієнтська частина системи автоматичного резервування авіаквитків.

Реалізацію було розбито на основні етапи: реалізація сторінки пошуку; реалізація сторінки квитка; реалізація сторінки пасажира. Кожен етап реалізації виконувався згідно вимог, зазначених у технічному завданні.

Дана система працює справно і виконує всі функції, які планувались.

РОЗДІЛ 4
ДЕМОНСТРАЦІЯ СИСТЕМИ АВТОМАТИЧНОГО РЕЗЕРВУВАННЯ АВІАКВИТКІВ

4.1 Демонстрація сторінки пошуку

Спочатку коли користувач заходить на сайт він бачить сторінку пошуку.

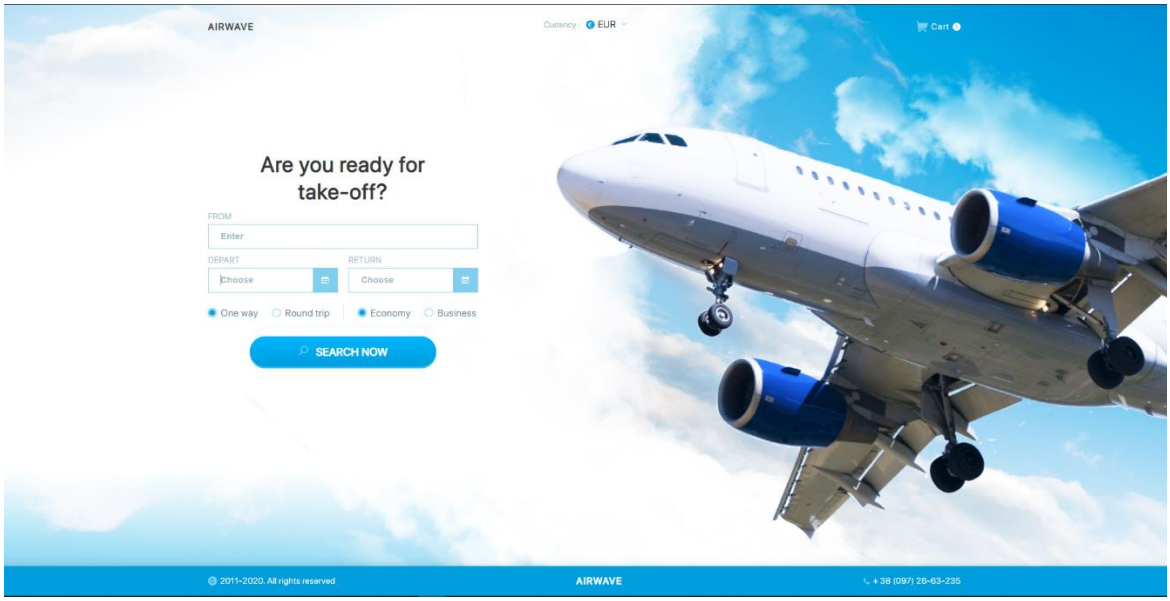


Рис. 4.1 Сторінка пошуку

Коли він починає вводити щось в пошук, йому показують всі можливі варіанти.

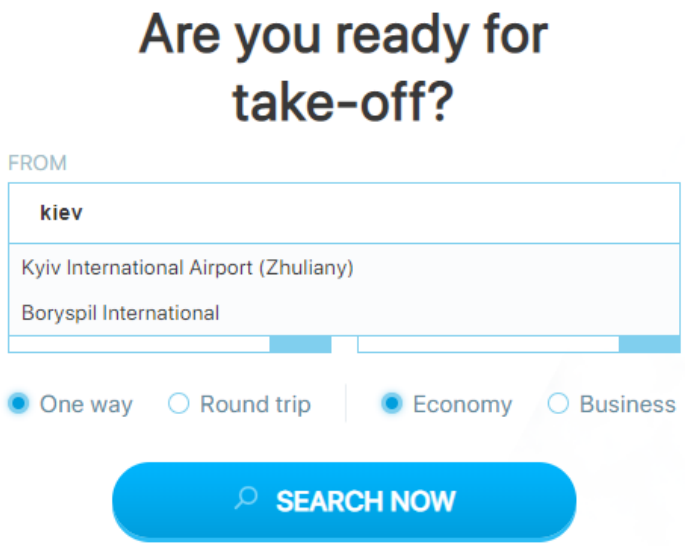


Рис. 4.2 Демонстрація пошуку міста

Введемо якісь дані щоб подивитись роботу наступних сторінк.

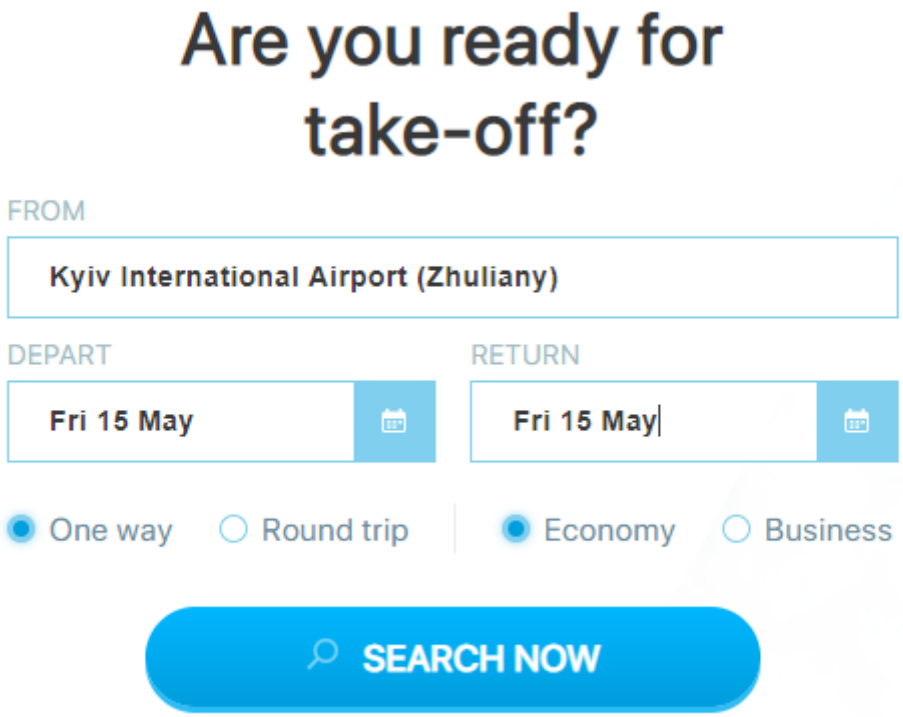


Рис. 4.3 Введені дані для прикладу

4.2 Демонстрація сторінки квитка

Після того як ми ввели дані і натиснули кнопку «SEARCH NOW», перед користувачем з’являється сторінка квитка, на якій можна вибрати підходящий квиток.

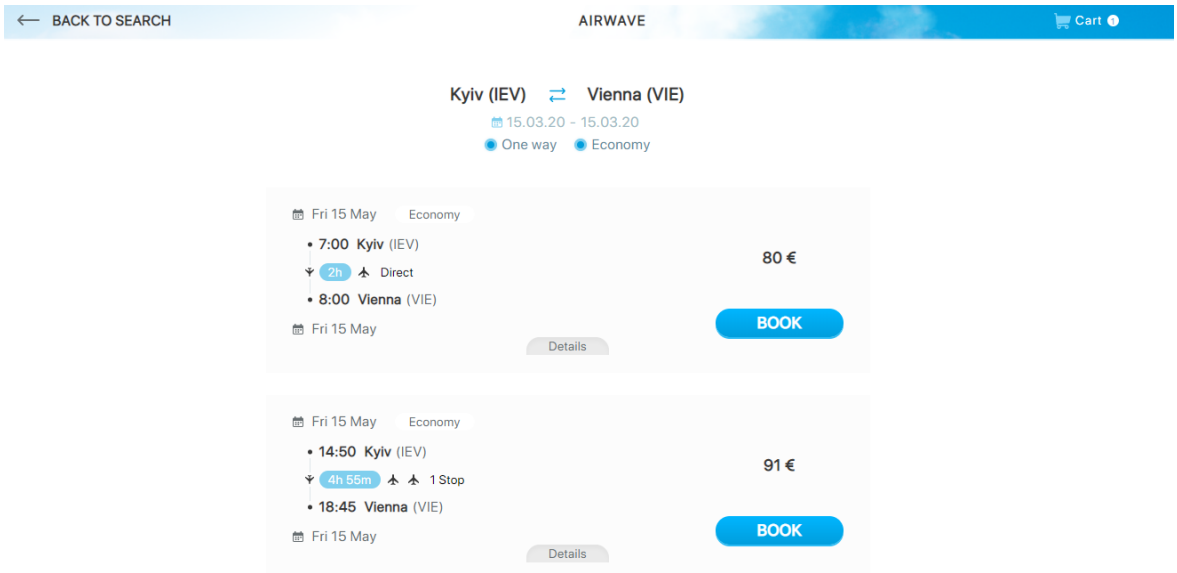


Рис. 4.4 Демонстрація сторінки квитка

Порівняємо з квитками, які нам видає kiwi.com.

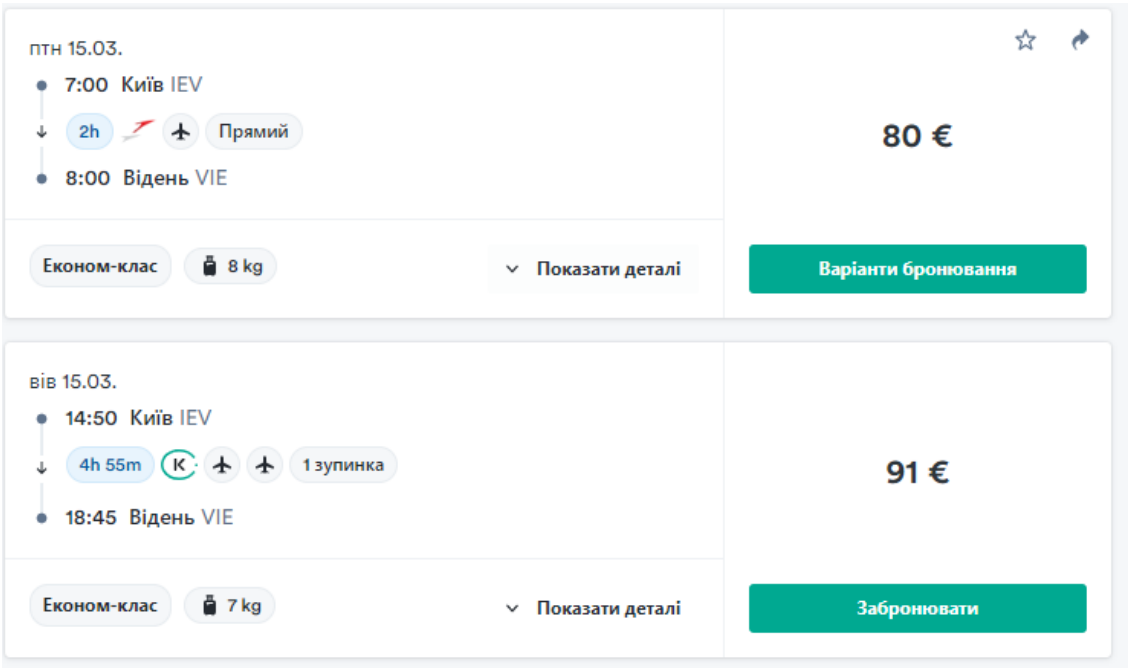


Рис. 4.5 Демонстрація результату Kiwi.com

4.3 Демонстрація модального вікна

Перевіримо чи правильні у дані в модальному вікні.

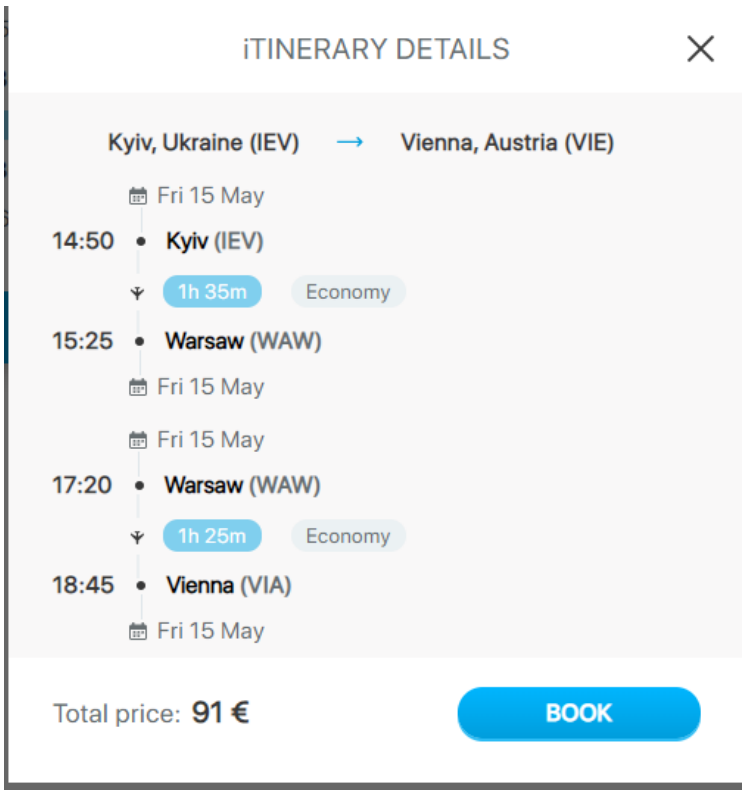


Рис. 4.6 Демонстрація модального вікна

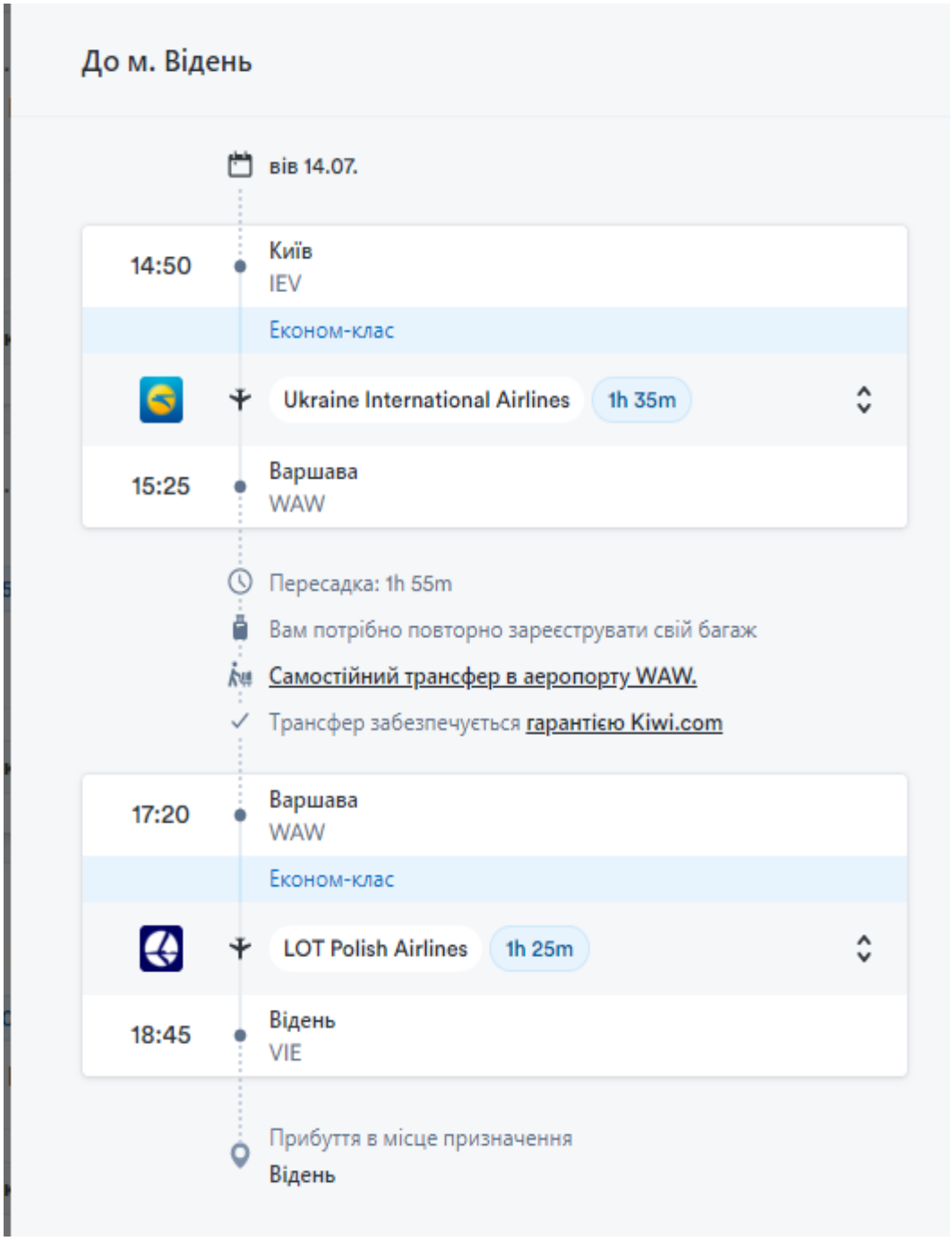


Рис. 4.7 Демонстрація модального вікна kiwi.com

Всі дані відображаються правильно.

4.4 Демонстрація сторінки пасажира

Після того як ми вибрали квиток і натиснули на кнопку «Book» ми переходимо на сторінку пасажира.

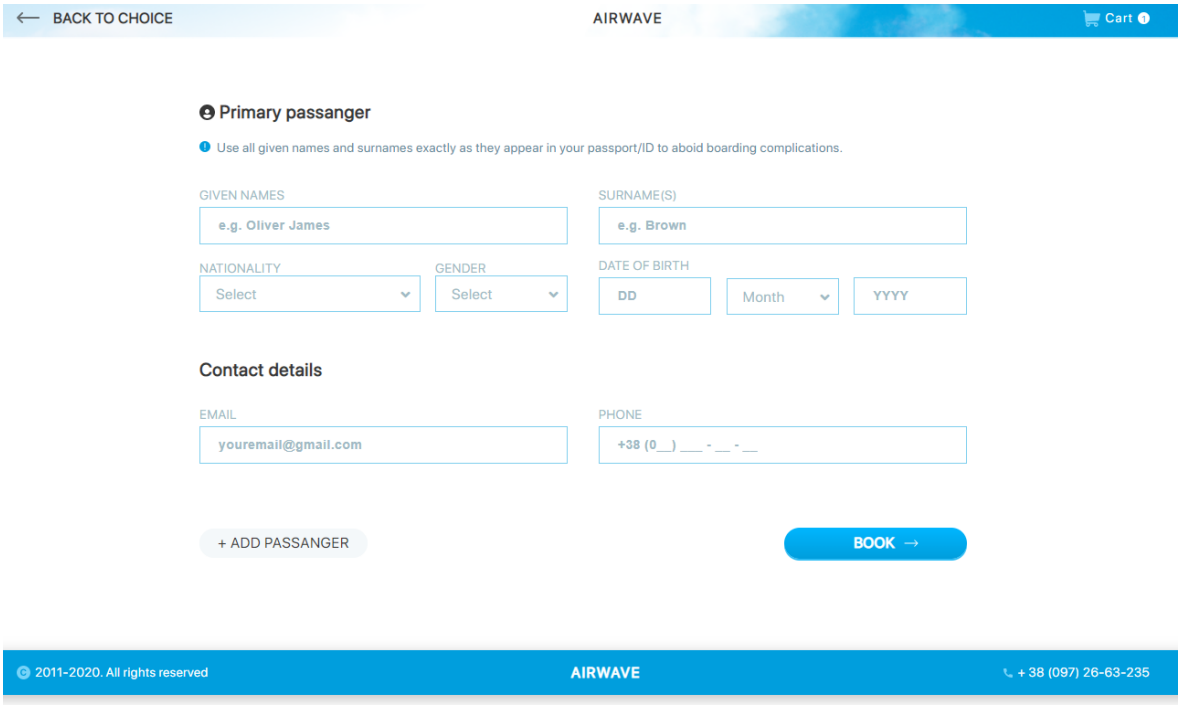


Рис. 4.8 Демонстрація сторінки пасажира

Перевіримо чи можливо додавати ще одного пасажира.

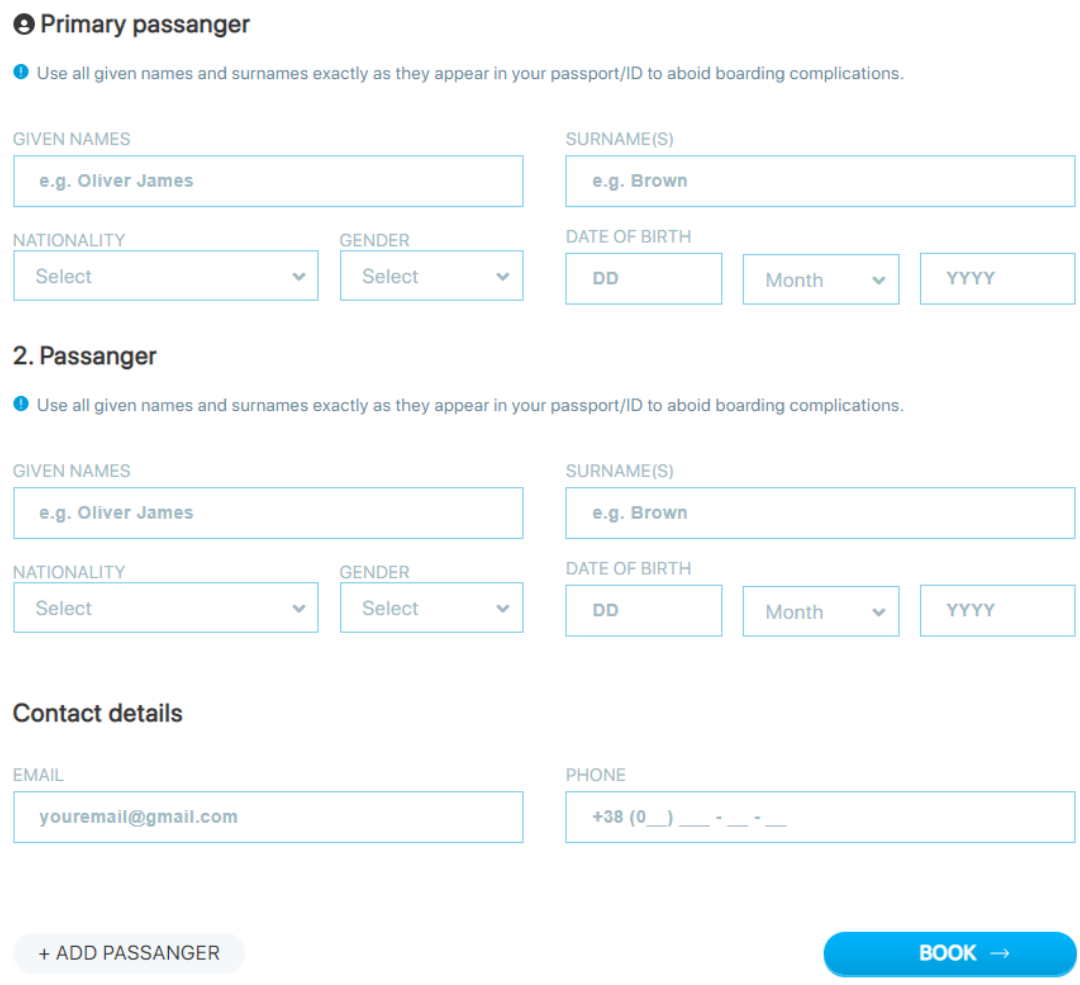


Рис. 4.9 Демонстрація додавання ще одного пасажира

ВИСНОВОК ДО РОЗДІЛУ 4

В цьому розділі було проведено демонстрацію роботи системи. Ми переконались, що все працює правильно. Користувач може без проблем вільно користуватись цією системою і в нього не виникне ніяких складнощів в процесі роботи з нею.

Висновки

Розробка даного дипломного проекту була присвячена створенню клієнтської частини системи автоматичного резервування авіаквитків для вбудовування її в сайт готелю.

У ході виконання проекту були розглянуті існуючі системи автоматичного резервування авіаквитків та виявлені їх недоліки, що стали причиною створення нової системи.

Також було проаналізовано предметну область даної роботи та визначено основні вимоги до розробки.

Зважаючи на вище задані вимоги був проведений аналіз використання існуючих технологій для реалізації системи. Веб-версія буде реалізована майже у всіх можливих браузерах та відображатися практично однаково. Також було обрано фреймворк для розробки та приведено обґрунтування доцільності його використання.

Реалізація відбувалася у 3 основних етапи з урахуванням зазначеного у технічному завданні функціоналу та вимог до розробки, а саме реалізацію системи автоматичного резервування авіаквитків на базі відкритого API.

Список використаної літератури

1. [Електронний ресурс] – Режим доступу до ресурсу: https://lpgenerator.ru/blog/2013/10/21/chto-takoe-html-korotko-o-glavnom/
2. [Електронний ресурс] – Режим доступу до ресурсу: https://wiki.rookee.ru/css/
3. [Електронний ресурс] – Режим доступу до ресурсу: https://ipipe.ru/info/javascript
4. [Електронний ресурс] – Режим доступу до ресурсу: https://web-creator.ru/articles/about_frameworks
5. [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/React
6. [Електронний ресурс] – Режим доступу до ресурсу: https://jetruby.com/ru/blog/vue-js-preimuschestva-i-nedostatki/
7. [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/AngularJS
8. [Електронний ресурс] – Режим доступу до ресурсу: https://stfalcon.com/ru/blog/post/why-use-angularjs-for-webapps
9. [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Babel_(transcompiler)
10. [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Прогрессивное_веб-приложение
11. [Електронний ресурс] – Режим доступу до ресурсу: https://vuex.vuejs.org/ru/
12. [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ESLint
13. [Електронний ресурс] – Режим доступу до ресурсу: https://monsterlessons.com/project/lessons/komponenty-v-vue

ДОДАТОК 1

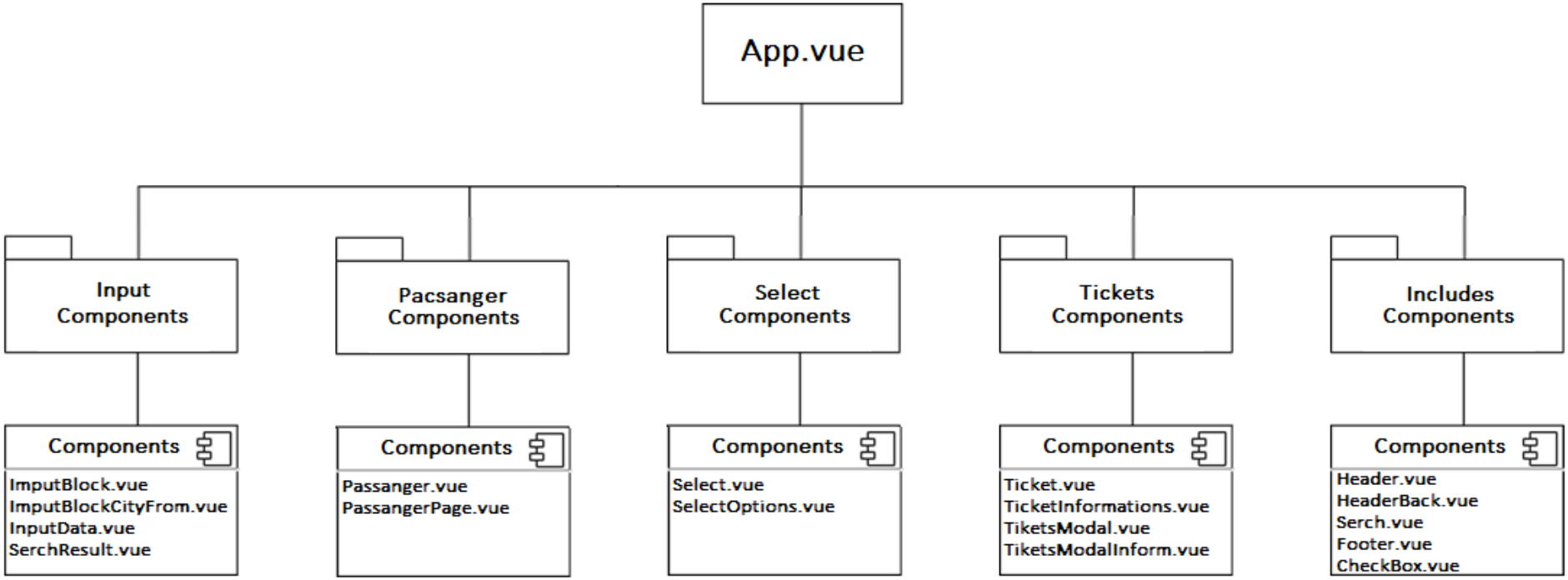
Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)

Схема структурна – структура програми

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ – 2020 р.



					ІАЛЦ.467100.004 Д1	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Падучак Д. В.*			*Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)*	*Літ.* / *Аркуш* / *Аркушів*
Перевірив		*Стешин В.В.*				*1* / *1*
Реценз.						
Н. Контр.		*Сімоненко В.П.*			*Схема структурна*	*НТУУ «КПІ», ФІОТ, ІО-63*
Затв.		*Стіренко С.Г.*				

ДОДАТОК 2

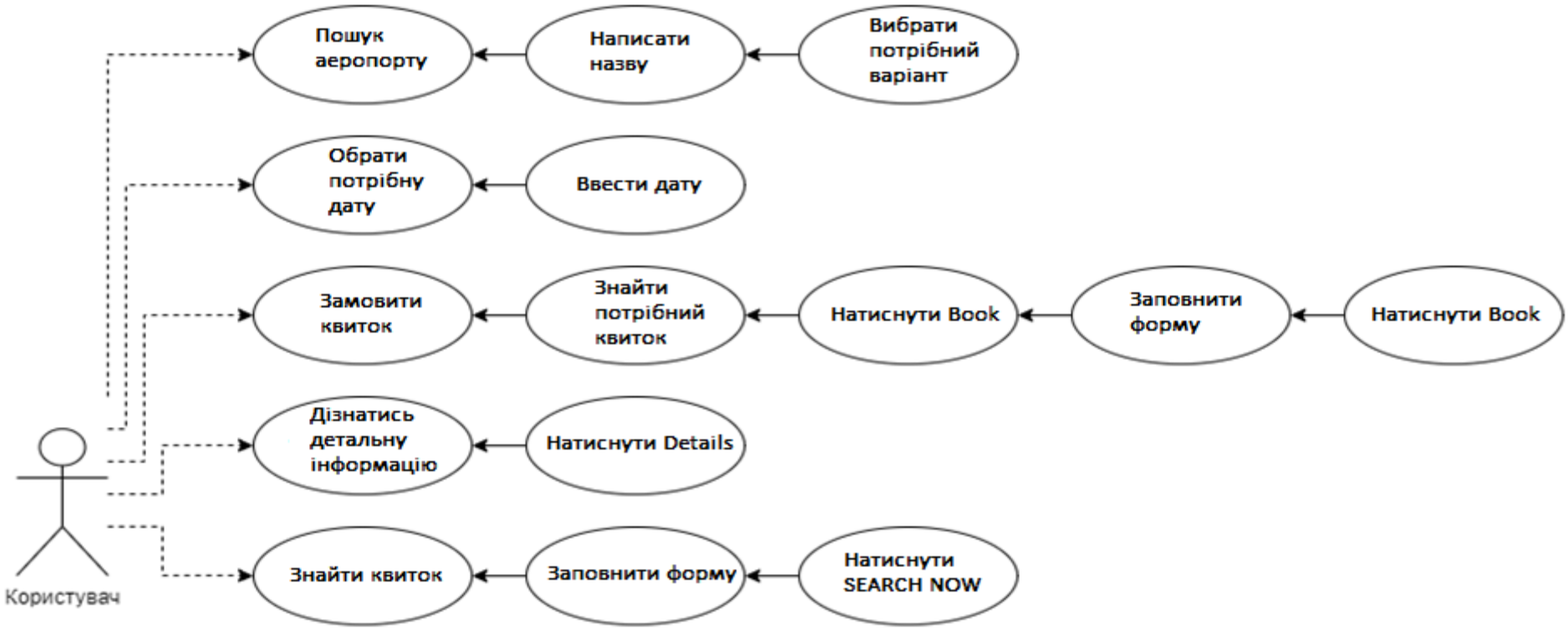
Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)

Схема функціональна – схема прецедентів

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ – 2020



					ІАЛЦ.467100.005 Д2	
Зм.	***Арк.***	***№ докум.***	***Підпис***	***Дата***		
Розробив		***Падучак Д. В.***			***Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)***	***Літ.*** / ***Аркуш*** / ***Аркушів***
Перевірив		***Стешин В.В.***				***1*** / ***1***
Реценз.						
Н. Контр.		***Сімоненко В.П.***			***Схема функціональна***	***НТУУ «КПІ», ФІОТ, ІО-63***
Затв.		***Стіренко С.Г.***				

ДОДАТОК З

Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)

Схема принципова – схема алгоритму пошуку аеропорту

ІАЛЦ.467100.006 ДЗ

Аркушів 1

Київ – 2020

Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.006 ДЗ		
Розробив		Падучак Д. В.			Система автоматичного резервування авіаквитків на базі відкритого API (клієнтська частина)	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Стешин В.В.					
Реценз.						НТУУ «КПІ», ФІОТ, ІО-63	
Н. Контр.		Сімоненко В.П.			Схема принципова		
Затв.		Стіренко С.Г.					